NASA-CR-189,543

NASA Contract Report 189543

NASA-CR-189543
19920011041

# MEASUREMENT OF VORTEX FLOW FIELDS

T. Kevin McDevitt, Todd A. Ambur, Gary M. Orngard, F. Kevin Owen

COMPLERE INC.
Palo Alto, CA

Contract NAS1-18667
January 1992



NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

## Abstract

The objective of Phase II was to design, build and demonstrate a three dimensional laser fluorescence anemometer for use in the Langley 16- by 24- Inch Water Tunnel. Innovative optical design flexibility combined with compact and portable data acquisition and control systems have been incorporated into the instrument. This will allow its use by NASA in other test facilities. The final instrument and support systems differ in several significant aspects from the design envisaged during our Phase II proposal preparations. Our original mirror traverse alignment concept has been replaced by a more versatile fiber optic system. This facilitates normal and off-axis beam alignment, removes mirror losses and improves laser safety. This added optical flexibility will also enable simple adaptation for use in the adjacent jet facility. New proprietary concepts in transmitting color separation, light collection and novel prism separation of the scattered light have also been designed and built into the system. Off-axis beam traverse and alignment proved much more complex than initially conceived. This led to the requirement for a specialized, programmable traverse controller and the inclusion of an additional traverse for the off-axis arm. To meet this challenge, an "in-house" prototype unit was designed and built and traverse control software developed specifically for the water tunnel traverse applications. A specialized data acquisition interface was also required. This was designed and built for the Laser Fluorescence Anemometer system.

## Introduction

At present, significant efforts are being made to effect design changes which will improve aircraft agility, maneuverability and performance. But, although significant progress has been made in computational aerodynamics, reliable design changes still cannot be made without recourse to experiment. Attempts to extend tactical flight envelopes still require extensive preflight ground based model testing. Unfortunately, conventional wind tunnel testing is expensive and time consuming and most facilities were built before present-day optical methods for quantitative flow field measurements were envisaged. Consequently, the non-intrusive detailed documentations of lee-side vortex flow-fields which are often required to support design evaluation and optimization are few.

However, in the past, qualitative water tunnel simulations have guided many practical designs and, since most of these facilities have been built with excellent optical access, they are ideally suited for use in advanced flow field diagnostics. Since the performance of most lifting and maneuvering bodies is governed by extensive transitional and turbulent viscous wakes and vortical lee-side flows, non-intrusive optical measurement techniques are required. Consequently, water tunnels offer the opportunity to obtain inexpensive, detailed flow field measurements to support "cut and try" designs and basic research.

To realize this capability, a two dimensional laser fluorescence anemometer was built and tested in the Ames-Dryden Water Tunnel during Phase I. The instrument was used in an experimental study of vortex flow fields designed to determine the mechanisms and feasibility of controlling vortex breakdown by introducing relatively low rates of jet blowing along the vortex core. The objective of Phase II was to build a three dimensional instrument for studies in the

Langley 16- by 24- Inch Water Tunnel.

## Background

When a slender delta wing is at an angle of attack to an oncoming stream, the upper and lower surface boundary layers flow outward and separate from the leading edges to form two free shear layers that roll up into a pair of vortices above the wing. Increasing angle of attack strengthens the vortices until the induced wing pressure field and associated adverse streamwise pressure gradients cause vortex breakdown. The flow is further complicated as the leading edge vortices mix with the wake from the trailing edge downstream of the wing. The phenomenon of vortex breakdown (or vortex bursting) can have a significant influence on control surface performance and unsteady loading. The inherent unsteadiness of the breakdown process compounds the problem as it continually moves the breakdown region back and forth along the vortex axis. This creates serious time dependent flow problems and asymmetrically disposed breakdown positions above the wing that are aggravated with side-slip.

Wide variations of breakdown patterns have been observed, and with increasing swirl the patterns change from spiral to near axisymmetric (Ref. 1). Spiral breakdown most commonly occurs over delta wings. In this breakdown process, the filament of fluid along the axis does not spread out symmetrically from a fixed stagnation point but, instead, takes on a spiral form around an unsteady "stagnation point" which varies in both space and time. Axisymmetric breakdown over delta wings, although rare, can also occur (Ref. 2). In this case, the vortex has a roughly axisymmetric breakdown pattern with a characteristic bubble which can have single or multiple tails (Ref. 3).

Unfortunately, the parameters and conditions that result in vortex breakdown are poorly understood, because reliable quantitative experimental data are difficult to obtain. With limited experimental information to guide flow field modeling, numerical studies of vortex breakdown and control have met with only limited success (Ref. 4). There have been two principal reasons for this. In the first place, flow field unsteadiness associated with breakdown produces directional intermittency. This leads to large uncertainties in mean and unsteady flow measurements obtained with conventional pitot and hot wire probes. Secondly, and perhaps more important, is the fact that vortex breakdown is known to be extremely sensitive to any form of introduced disturbance. Probes, due to their blockage, may drastically alter the breakdown position. For these reasons, almost complete reliance has been placed on flow visualization techniques to determine flow field characteristics. In the past, air or hydrogen bubbles have been used as tracers to visualize flow patterns in water tunnels. For steady flows, streak lines can be identified with streamline patterns. However, in more complex flows of practical interest, the use of bubbles for flow visualization has distinct drawbacks. First of all, their introduction acts as a fluid lubricant which alters the apparent fluid viscosity and so its turbulent structure. Secondly, light refraction at the gas/water interfaces will destroy laser beam coherence and make it impossible to obtain laser velocimeter measurements in the regions where the tracer is present. But, with the advent of the laser fluorescence anemometer, there are now opportunities to determine accurate quantitative flow field velocity measurements of the vortex bursting process.

## The Laser Fluorescence Anemometer

The principle of operation of the laser fluorescence anemometer is shown schematically in Fig. 1. The mean velocity and turbulence measurements are made with a dual-beam velocimeter utilizing a Bragg cell that enables moving interference fringes to be generated in the focal volume so that instantaneous velocity magnitude and direction measurements can be achieved from the frequency shift ($f_D$) around the incident and modulated laser beam interference frequency ($f_0$). i.e. $U = \lambda(f_D - f_0) / 2\mathrm{Sin}\ (\theta/2)$ where $\lambda$ is the wavelength of the incident laser light. Mean and fluctuating concentration measurements are achieved by observing the intensity of fluorescent light emitted from the focal volume at a different wavelength ($\lambda_C$). At correct levels, tracer fluorescence is linearly proportional to the trace material concentration and, therefore, fluorescent intensity is directly proportional to the concentration of fluid from the seeded flow in the focal volume. The cathode current from a second photomultiplier tube is coupled to a high-gain current to voltage converter to produce a continuous voltage proportional to the instantaneous concentration.

Since fluorescence is such a complex phenomenon dependent upon many parameters, only those of particular importance to the present application have been considered. In this context, the principal requirements were linearity combined with adequate sensitivity (i.e., signal/noise ratio), frequency response, and spatial resolution. Since the fluorescent intensity of a particular organic dye can be a strong function of the solvent, which in this case was room temperature water, other dye-solvent-temperature combinations would produce different ( and possibly increased) fluorescence. However, the fluorescent output was more than adequate for the present tests. In addition, the relationship between fluorescent intensity and concentration is of course exponential but, at the extremely low dye concentrations used in these experiments, a linear approximation could be made without introducing significant errors.

The data in Fig. 2 show this linearity of the present technique, which employed dysodium fluorescein dye (a sodium salt of fluorescein which has been used for flow visualization for many years) as the trace material in water. Since, for a given dye concentration, the measured fluorescence is a function of laser beam intensity and collection optics, the ordinate of Fig. 2 has been plotted in arbitrary (voltage) units. The high signal-to-noise ratio that can be obtained for dye concentrations as low as 0.04 ppm (by weight) is illustrated in Fig. 3. The rise time ($\approx$ 50 $\mu$sec), which corresponds to the time taken to chop the laser beam, shows the adequate frequency response of the system.

Since there is usually an overlap of the absorption and emission curves for most dye-solvent combinations it is possible that fluorescent photons emitted from the probe volume could be reabsorbed by the dye molecules that are between the probe volume and the collection optics. This effect could produce a range dependent signal. However, fluorescent intensity measurements previously obtained across an entire test section when filled with stagnant water at maximum dye concentration showed that these effects and those of possible beam attenuation were negligible. Thus, unlike absorption techniques which measure integrated ("line of sight") properties, the receiving optical arrangement primarily governs the spatial resolution of the fluorescence technique. In the present experiments, off-axis light collection and multimode fiber aperture size resulted in a maximum focal volume length dimension of approximately 0.5 mm although smaller

spatial resolution can be achieved by appropriate choice of collection optics without affecting the LDV, since the velocimeter interference fringes are moving.

The instrument delivered to NASA LaRC comprises three primary elements namely: the optical, traverse control, and data acquisition systems. The Laser Optical system (Fig. 4) uses a 6 watt Argon-ion laser. The transmitting optical arrangement (Fig. 5) is straightforward with a few unique features addressing the common problem of beam distortion or thermal blooming at higher laser powers. Frequency shifting is done before the color separation prisms using a single Acousto-Optic modulator made of a selected flint glass which can handle the full laser power with minimal distortion. (For significantly greater laser powers we also found a water cooled Bragg cell which was more than required for this application.) This is followed by the color separation prisms, the first of which are fused silica for power handling capacity. A final prism of dense flint provides maximum dispersion once the laser beam has been split into at least eight beams. Final color selection is made using right angle prisms. The lines used for this experiment were 514.5nm, 488nm, 476.5nm. Although the emission spectrum of the fluorescence is centered about 515nm, there was sufficient higher wavelength emission for the edge filter to select wavelengths above 525nm. Other laser lines could have been selected if needed for fluorescence excitation or for separation from emission lines.

Pure fused-silica core single-mode polarization-preserving fibers are used for light transmission; two fibers per color. The use of optical fibers not only avoids the tedium of mirror-traverse alignment, but also greatly simplifies the transmission of light to the third axis. The pure fused silica core fibers seem immune from the progressive transmission losses which are found in other fibers. Polarization preserving fibers provide greater modal stability when the fibers are flexed or manipulated. For mechanical protection the fibers are armored and contained within a conduit. Upon exiting the fibers the beams are collimated at 4.4 mm dia. with a separation of 60mm.

Forward-scattered light is collected with a single 80mm diameter lens and focused into a 200 μm multi-mode optical fiber which conducts it to the color separation and signal detection box (Fig. 6). The fluorescence signal is split off with an edge filter. For maximum efficiency, a prism separation scheme is used rather than di-chroic filter and interference filters for the LDV signals.

Experience has shown that accurate positioning is vital to a successful test program. Accurate positioning is complicated in an air-window-water environment due to the difference in refractive indices on either side of the water tunnel window. Refractive index problems are particularly acute for the third component beams which are transmitted at 45 degrees to the normal incidence of the four beam axis. Fig. 7 illustrates the problem, which may not be intuitively obvious. In order to traverse the measurement volume horizontally some distance in water, the two orthogonal optical components, axial(x) and model vertical(z), must be traversed some lesser amount in air. In order for the third or off-axis pair of beams to intersect the measurement volume at the required angle of thirty degrees, the beams must strike the window at an angle of approximately forty-five degrees. In addition, linear horizontal movement of the focal point of the third pair of beams requires lens movement on a sloping line. As the lens gets closer to the window, it will have to rise. The length of movement on this sloping path will be only about half

of the movement of the focal point in water. Thus, two lens systems must move on different paths at different speeds in order to maintain a coincident focal point. To sort this all out, two three-axis traverse tables were installed for computer controlled, algorithm driven positioning of the LDV probe volume and forward scatter collecting optics. A fourth traversing axis supports the off-axis transmitted beams. Position is maintained by a custom designed eight axis traverse controller with micro-stepping drives, optical encoder feedback, and limit switch safety stops. Details of the Traverse Control System are given in Appendix A.

A Laser Velocimeter Data Acquisition System (LVDAS) has been designed. This instrument processes one to three channels of LDV data and digitizes up to four channels of analog data, one of which represents the concentration of dye. The instrument ensures coincidence and multiplexes the data to the computer. Velocities and analog channel values are displayed as well as data rates. Details are given in Appendix B.

Fig. 8 shows the modified data handling system for the 3D laser fluorescence anemometer. The continuous though not necessarily non-zero output from the high-gain current to voltage converter is fed directly into an analog to digital converter to provide 12 bits of digital information at 50 kHz. In water flows, this was more than sufficient to provide essentially real time point concentration data in digital form. But data from the three component LV system were not continuous wave since particle arrival times in the focal volume were random. However, whenever valid and essentially coincident data were received on all LV channels, a necessary requirement for shear stress measurement, these velocities, along with the instantaneous concentration voltage, was recorded. From a series of such readings, mean and turbulent velocity and concentration profiles were determined along with the turbulent shear stress and velocity/concentration cross correlations. These latter cross products provide new information on turbulent mixing rates in complex flows. In addition, we are able to determine details of the concentration/turbulent intermittencies from ensemble averages generated for selected instantaneous concentration levels. This will shed quantitative light on the turbulent structure and entrainment of fluid originating from different points in the flow field.

## Experimental Details

### *Test Facility*

The NASA Langley 16- by 24- Inch Water Tunnel is shown in Fig. 9. The tunnel has a vertical test section with an effective working length of about 4.5 ft. The velocity in the test section can be varied from 0 to 0.75 ft/sec., resulting in unit Reynolds numbers from 0 to $7.73 \times 10^4$ ft$^{-1}$ based on a water temperature of 78°F. The normal test velocity yielding smooth flow is 0.25 ft/sec, resulting in unit Reynolds numbers of $2.58 \times 10^4$ ft$^{-1}$ at 68°F. The model support system has deflection ranges of ±33° and ±15° in two planes of rotation. Rotation is accomplished via electronic remote control, and visual indicators allow the user to set angles within about ±0.25°.

The fluorescence seeding method for this investigation used fluorescein dye injected into the jet flow field from inside the model. Naturally-occuring particles in the water were used for seeding for the LDV part of the instrument. A representative size distribution provided by NASA is shown in Fig. 10.

*Model*

The test model selected by NASA was a non-axisymmetric afterbody propulsion model and is shown in Fig. 11. This model is a scaled-down version of a model to be tested in the NASA Langley 16-foot Transonic Wind Tunnel as part of a computational fluid dynamics code validation study. It can be used to simulate nozzle exit velocity ratios typical of those in the wind tunnel study. It consists of a generic forebody with a non-axisymmetric boattail and nozzle. Water is injected into the interior of the model and exhausted through flow-conditioning foam ahead of the throat and exits through the nozzle. Fluorescein dye is introduced upstream of the model in the water supply tube for the jet.

*Sample Test Results*

Unfortunately, optical beam refraction problems caused by complex test section wall deformations (≈0.10") under hydrostatic loading impeded laser beam alignment although limited data were obtained at zero angle of attack and a nominal jet exit to free stream velocity ratio of 1.5. For these test conditions, the cross flow velocity components (v,w) were negligible as expected. However, the model did provide a flowfield in which the basic instrument concepts could be verified. Fig. 12 shows the measured axial velocity distributions. It shows the extent of the jet and the location of the velocity defects in the model wall wakes. Clearly, the jet is highly skewed, but this has since been corrected by subsequent modification of the internal model flow treatment devices. The mean concentration profile (F
g. 13), which is much more symmetric, defines the extent of the jet at this axial station. Some insight into the unsteady features of the flow can be obtained from the fluctuation measurements. Figs. 14 and 15 show there is significant mixing in the outer shear layers between the jet and freestream flows. Peak fluctuation levels in the regions of maximum mean gradients indicate that small scale mixing is dominant. A measure of the level of streamwise mixing in the jet can be determined from Fig. 16. As expected the $\overline{u'c'}$ cross correlation function is positive as faster moving jet fluid is associated with higher concentration whereas fluid originating from the slower moving freestream has lower or zero concentration.

## Concluding Remarks

A Laser Fluorescence Anemometer which comprises a three component laser doppler velocimeter system with a fourth channel to measure fluorescent dye concentration has been installed in the NASA Langley Water Tunnel. The system includes custom designed optics, data acquisition, and traverse control instruments and a custom software package.

Feasibility studies clearly demonstrate how water tunnels can be used in conjunction with advanced optical techniques to provide non-intrusive detailed flow field measurements of complex fluid flows with a minimum of expense. The measurements show that the laser fluorescence anemometer will provide new insight into the structure, entrainment and mixing of vortical and shear layer flows.

## References


1. Sarpkaya, T., *Vortex Breakdown in Swirling Conical Flows* AIAA Journal, Vol. 9, No. 9, p. 1792, 1971.

2. Lambourne, N. C. and Bryer, D. W., *The Bursting of Leading Edge Vortices--some Observations and Discussion of the Phenomenon.* Brit. A.R.C.R. & M., No. 3282,1962.

3. Faler, J. H. and Leibovich, S., *Disrupted States of Vortex Flow and Vortex Breakdown.* The Physics of Fluids, Vol. 20, No.9, p. 1385, 1977

4. Karashima, K. and Kitama, S., *The Effect of a Small Blowing on Vortex Breakdown of a Swirling Flow.* Computational Techniques & Applications: CTAC 83, p. 553, 1884.

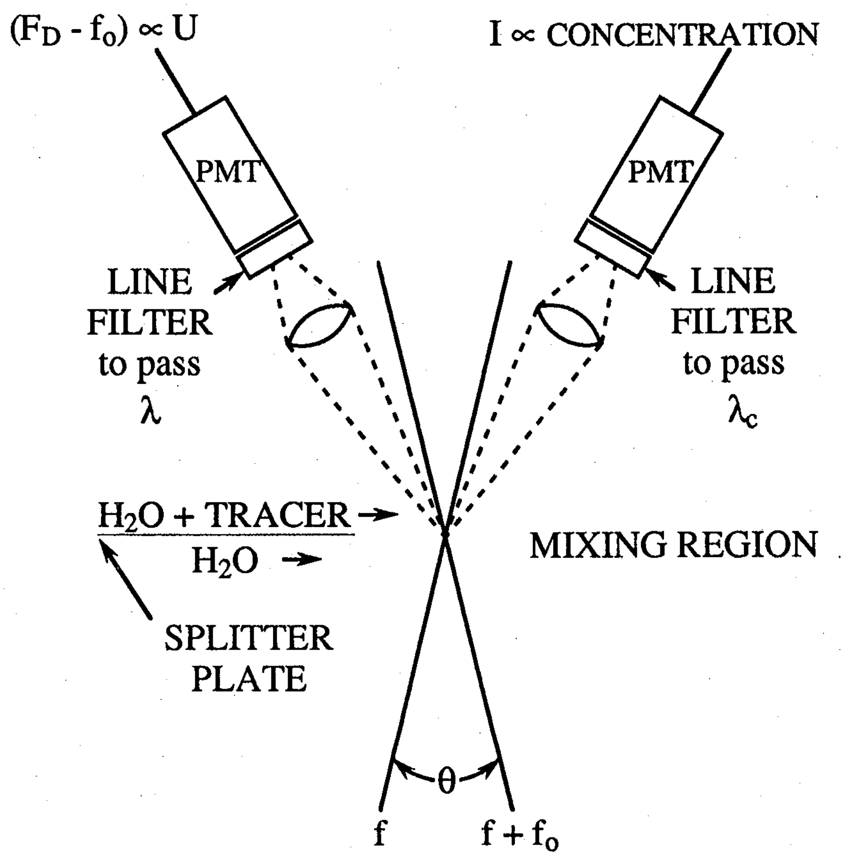




Fig. 1 Laser Fluorescence Anemometer

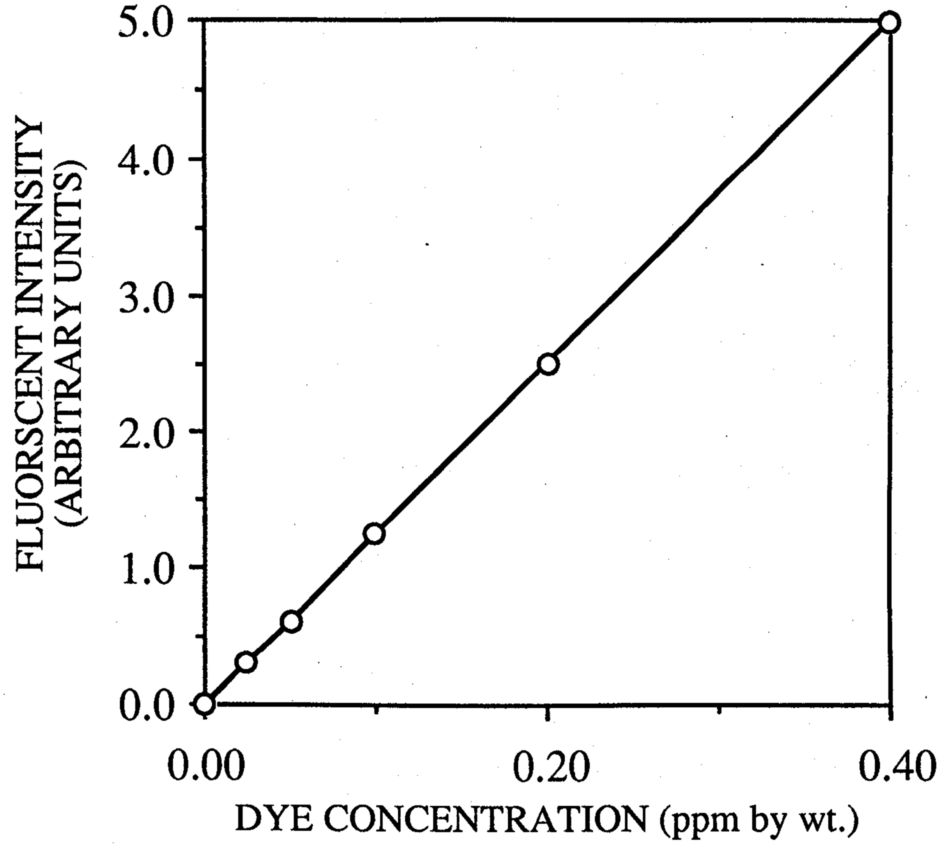



Fig. 2 Relationship Between
Dye Concentration and Fluorescent Output

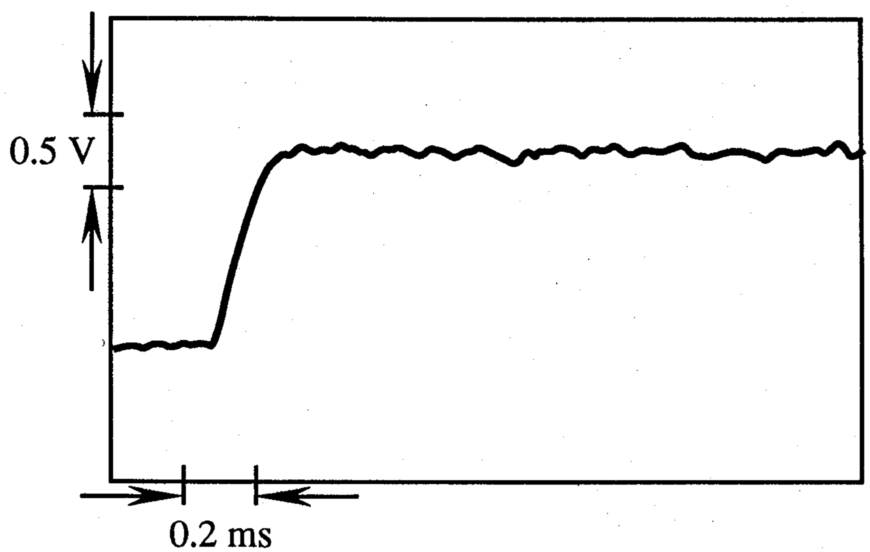



Fig. 3 Fluorescence Sensitivity and Frequency Response

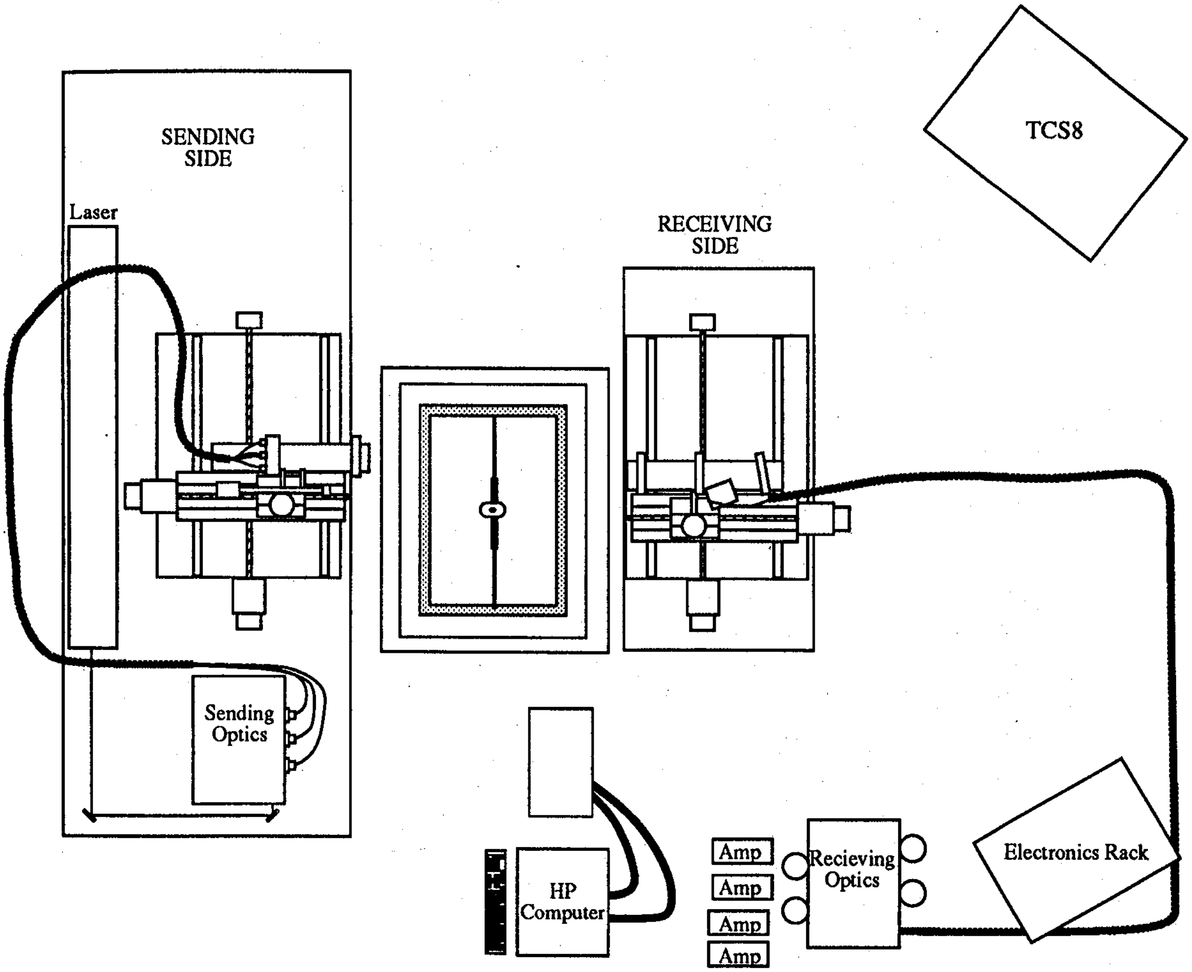



Fig. 4a Schematic of Plan View LFA System

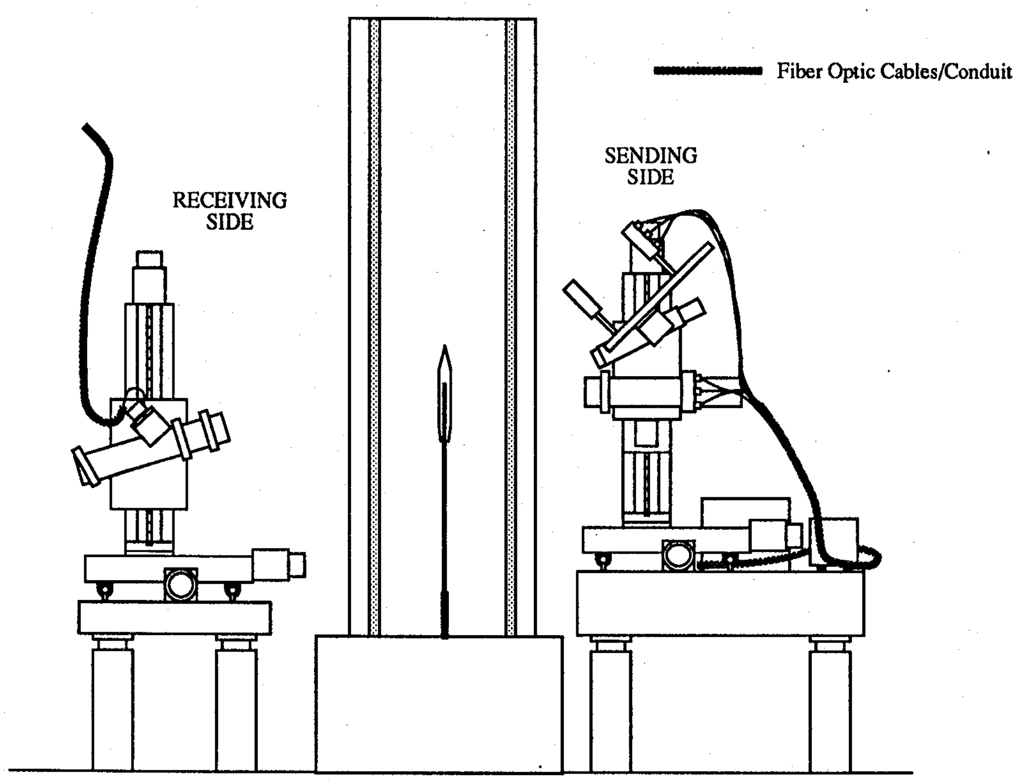



Fig. 4b Schematic of Side View LFA System

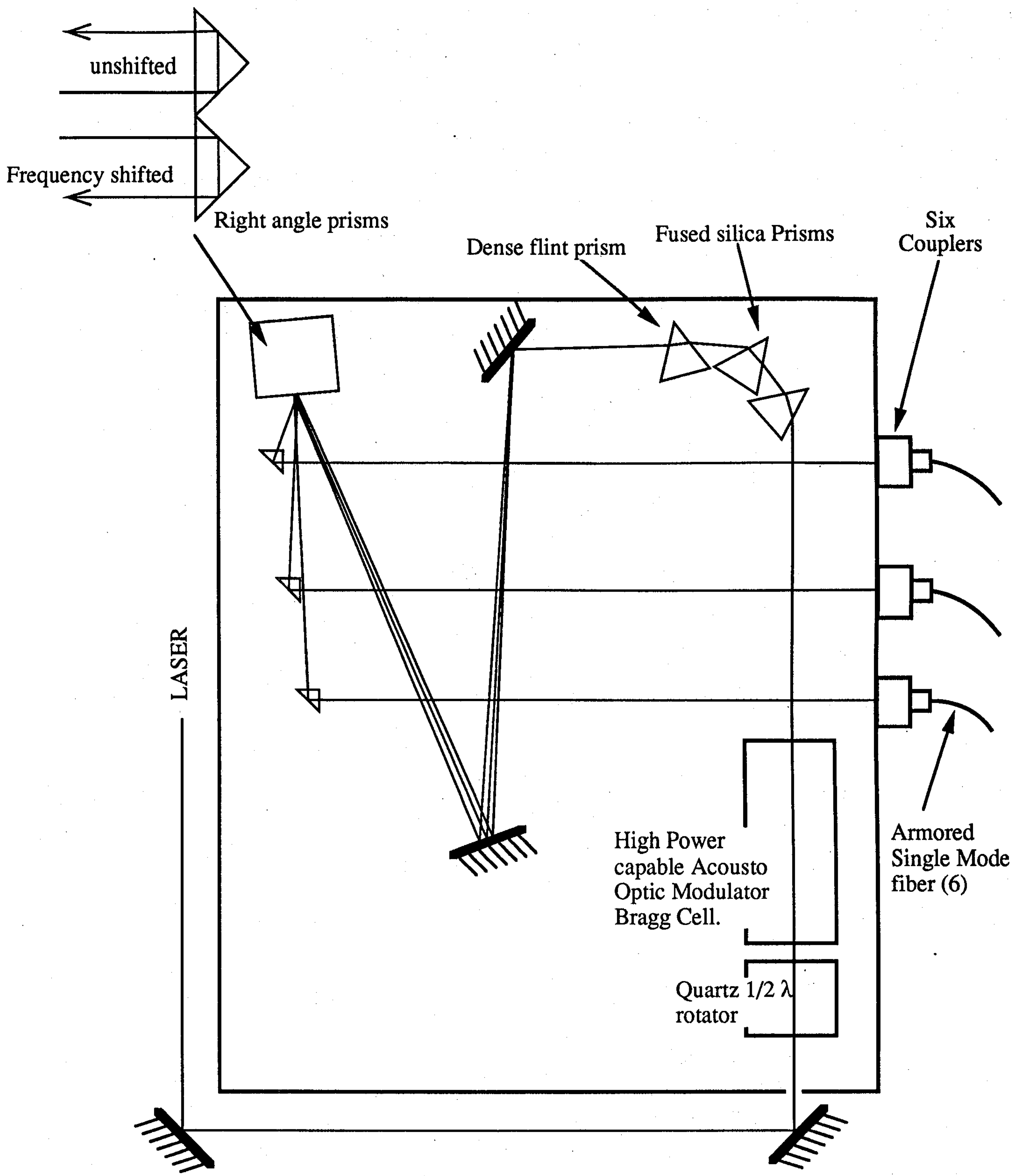



Fig. 5 Sending Optics

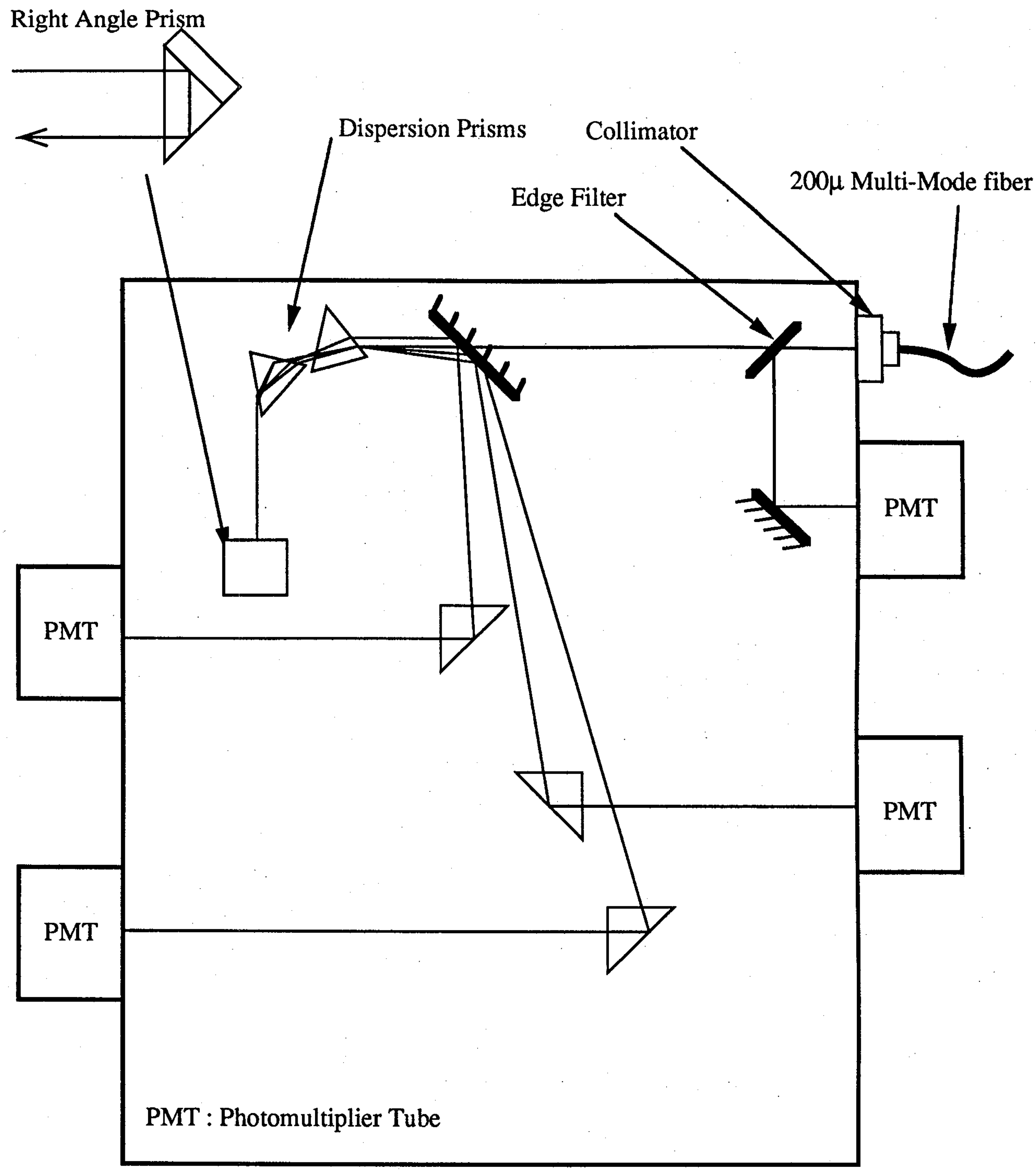
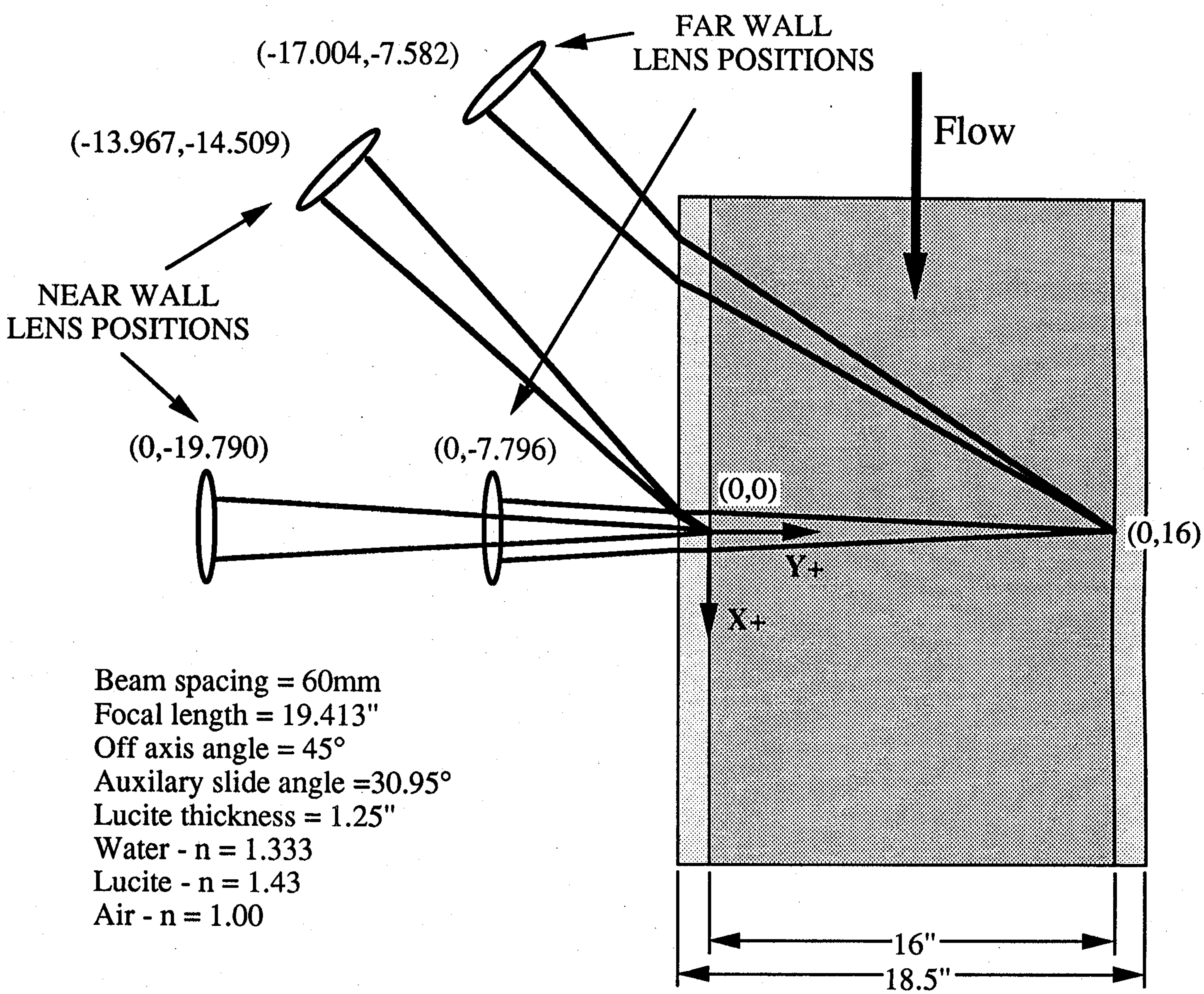



Fig. 6 Receiving Optics




Beam spacing = 60mm
Focal length = 19.413"
Off axis angle = 45°
Auxilary slide angle =30.95°
Lucite thickness = 1.25"
Water - n = 1.333
Lucite - n = 1.43
Air - n = 1.00

Fig. 7 The Refraction Problem

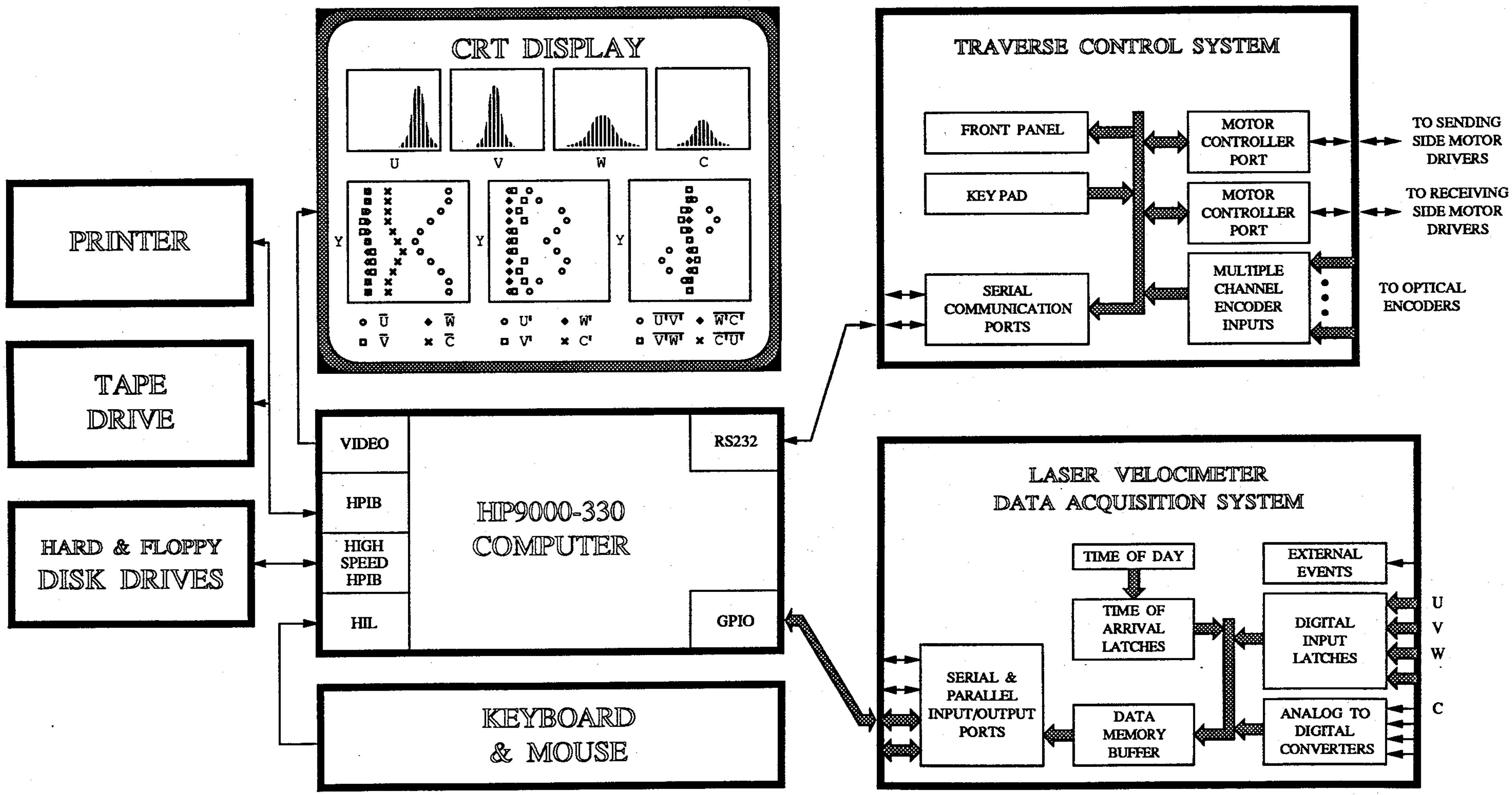



Fig. 8 Data Acquisition and Traverse Control Systems

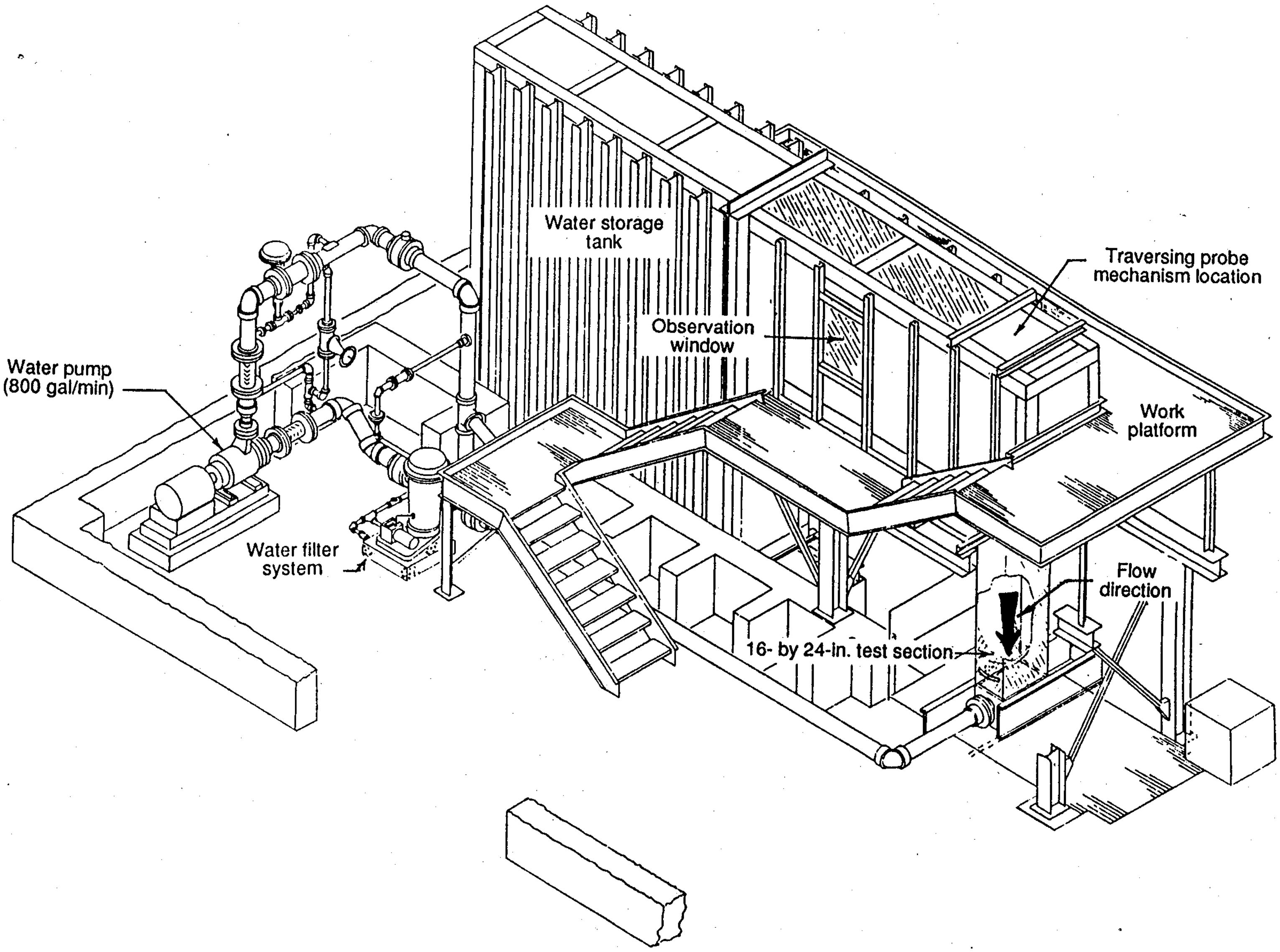



Fig. 9 Langley 16- by 24- Inch Water Tunnel

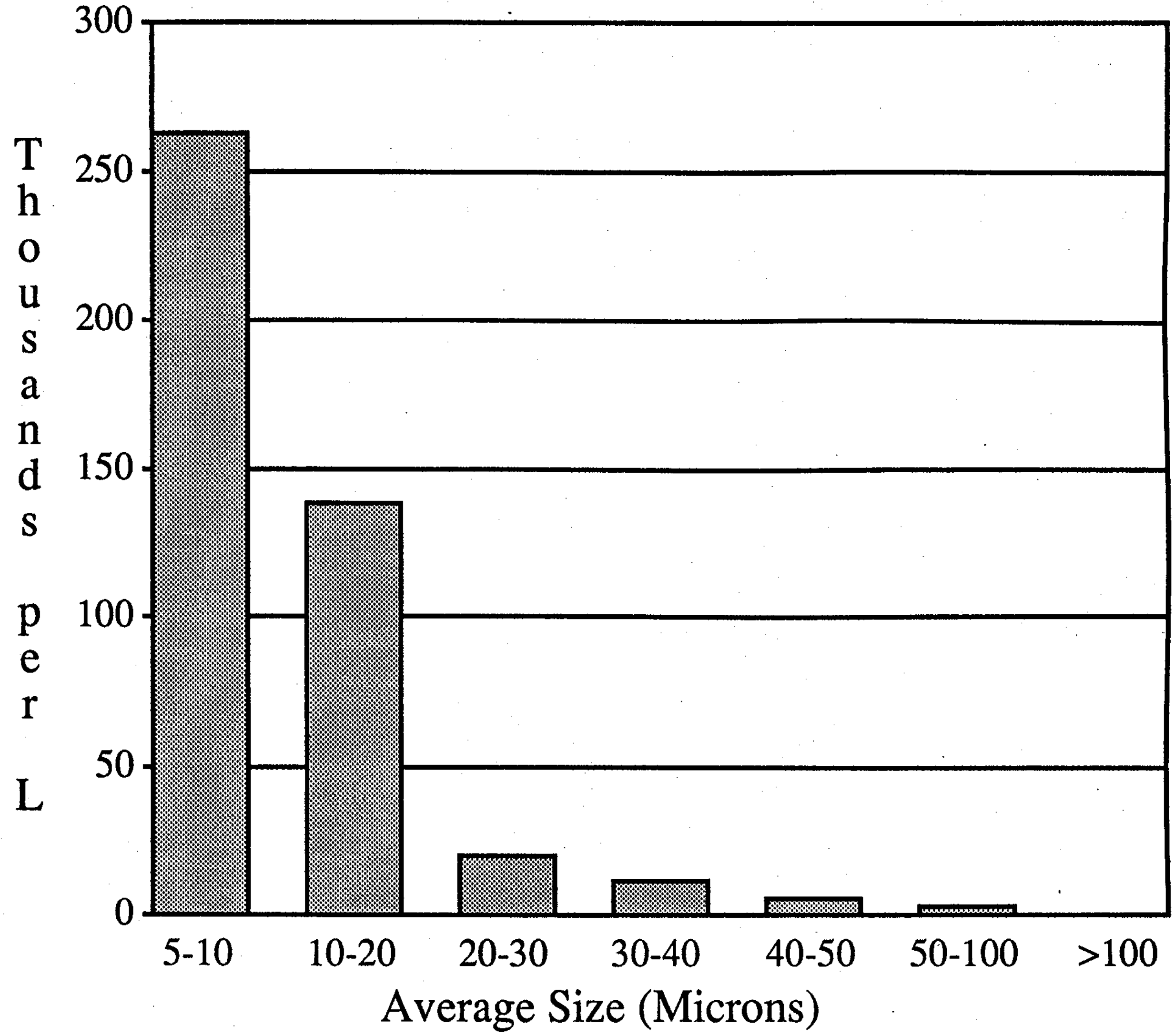



Fig. 10 Size Distribution Particle Concentration

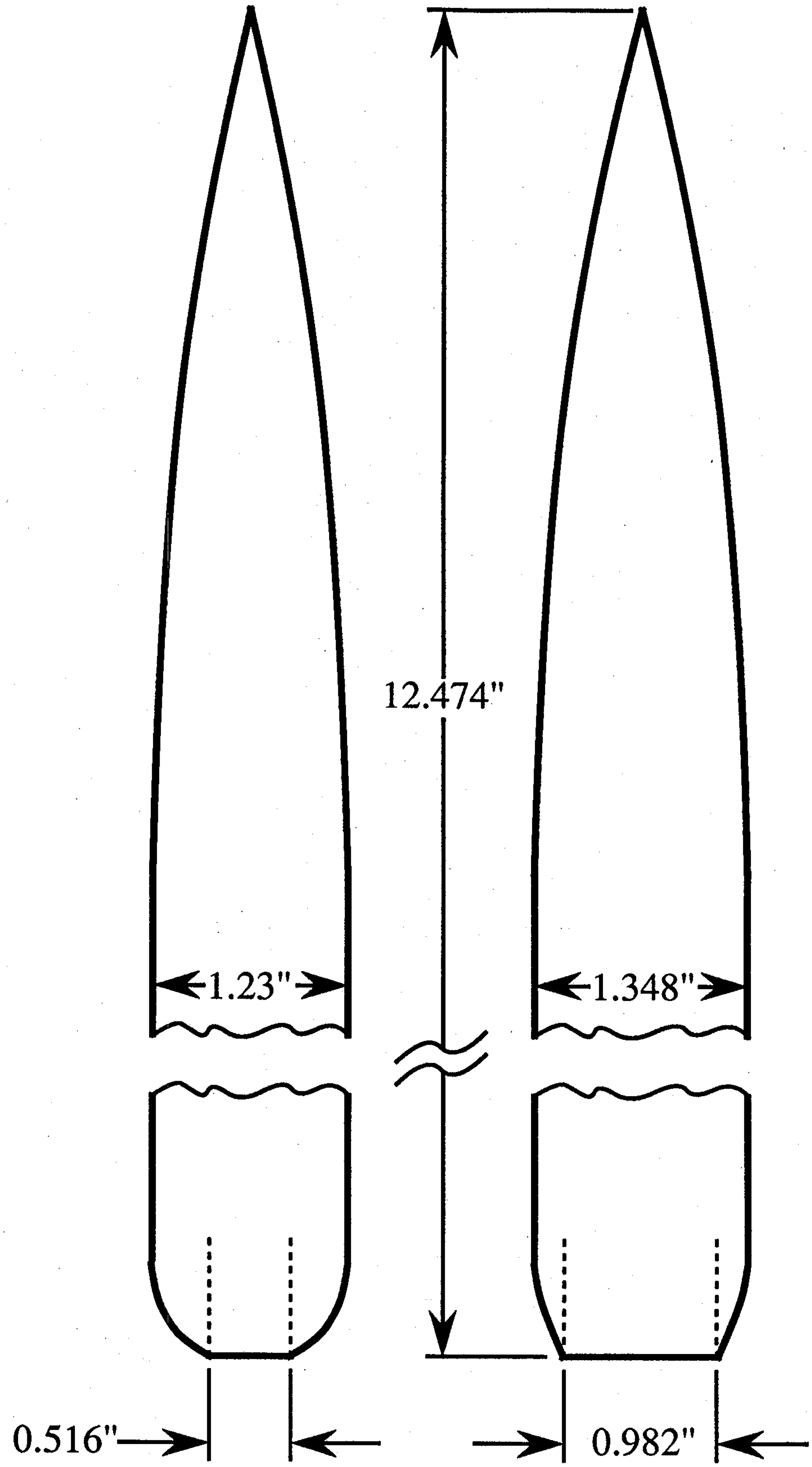



Fig. 11 Propulsion Model

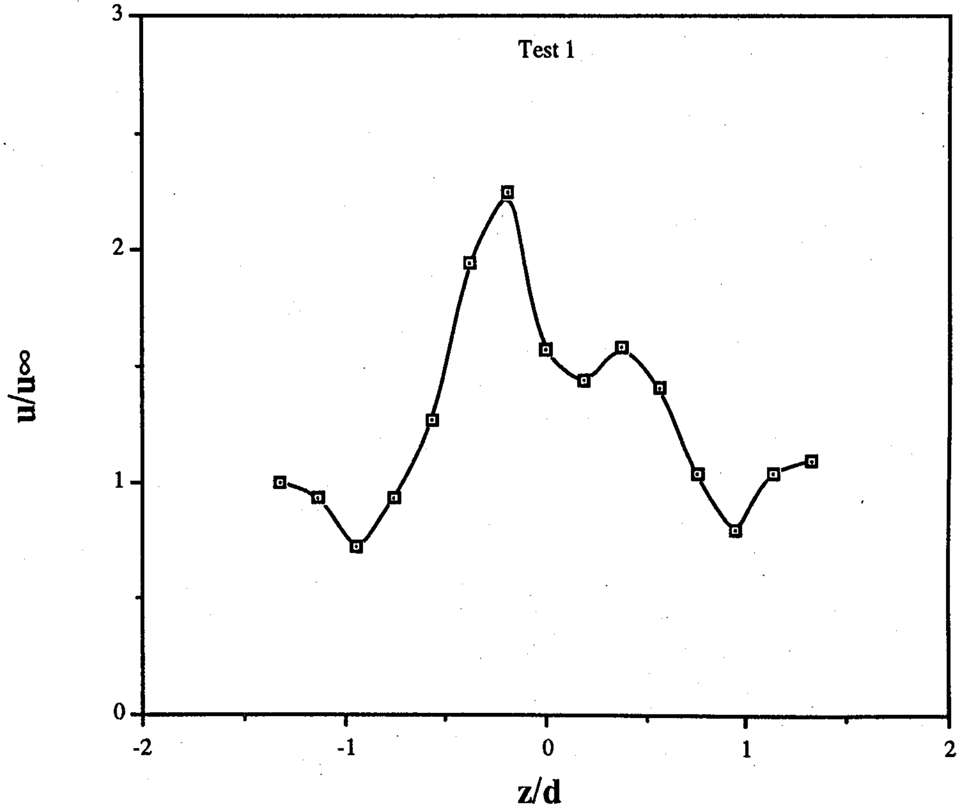



Fig. 12 Axial Velocity Profile

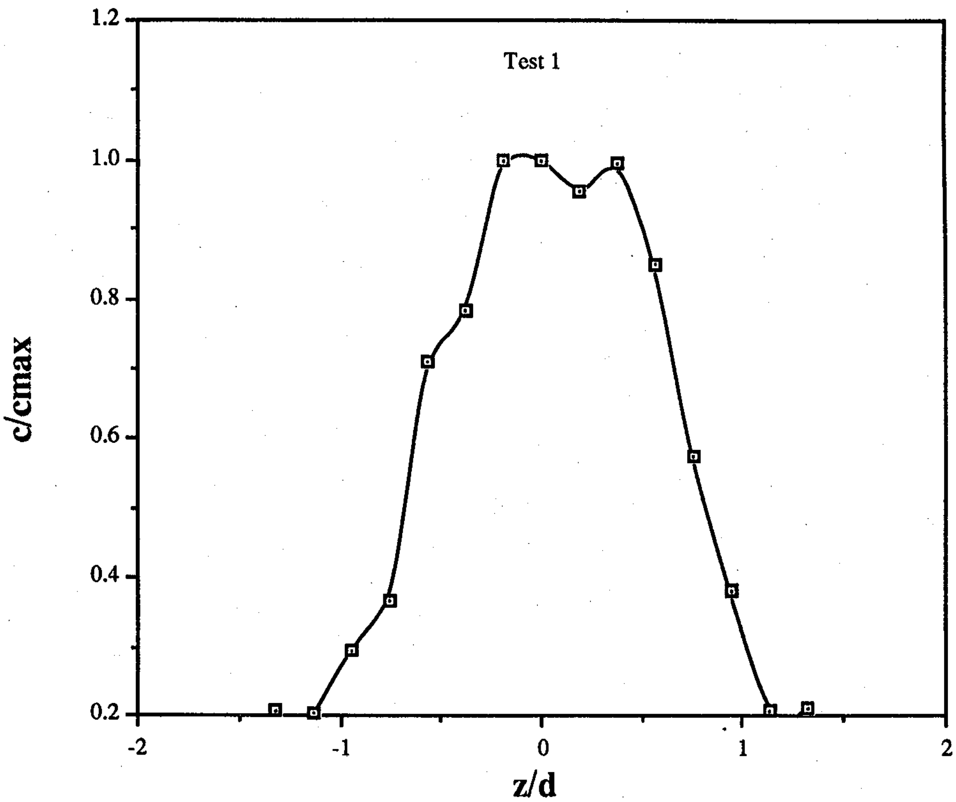



Fig. 13 Concentration Profile

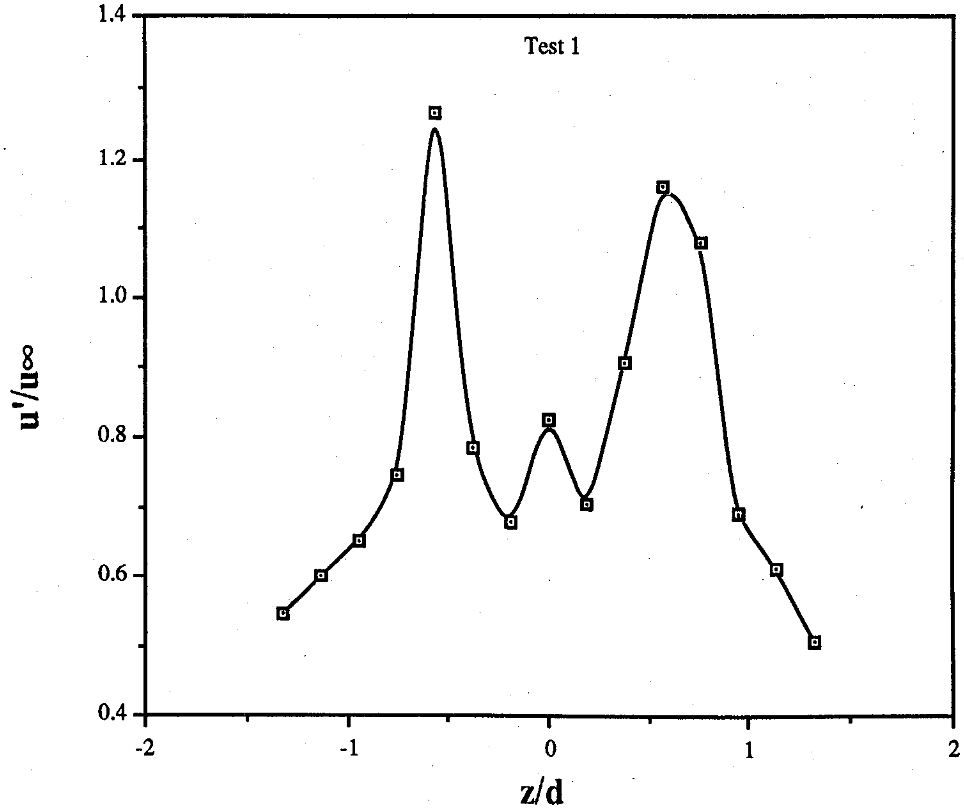



Fig. 14 Velocity Fluctuations

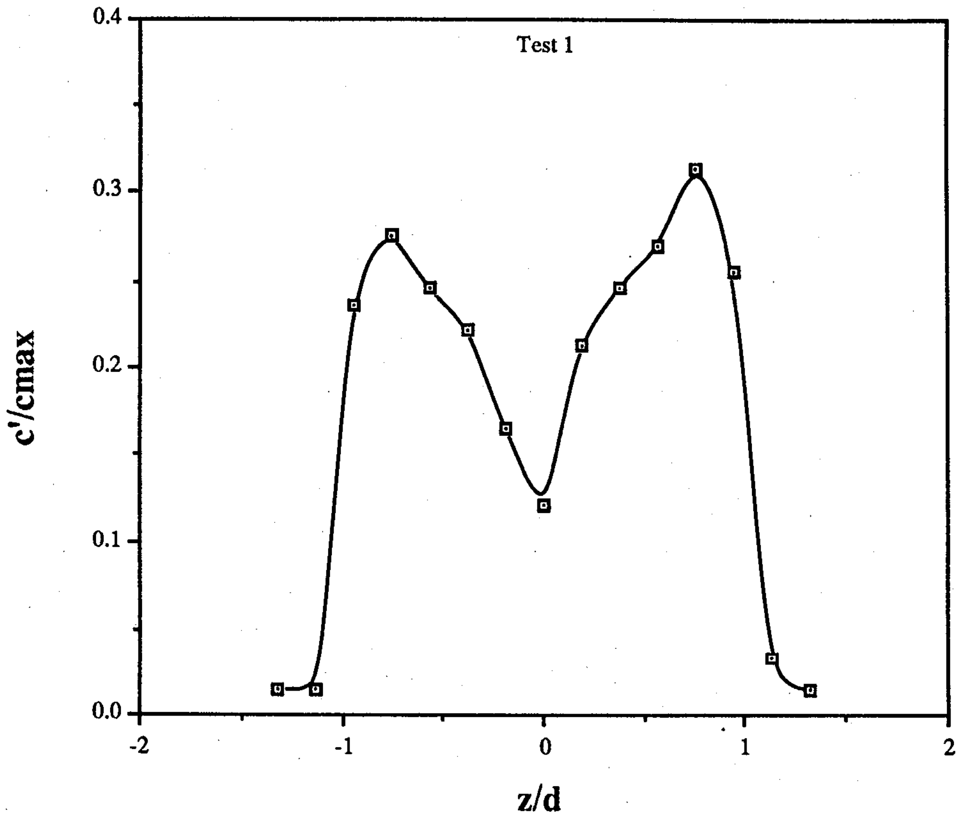



Fig. 15 Concentration Fluctuations

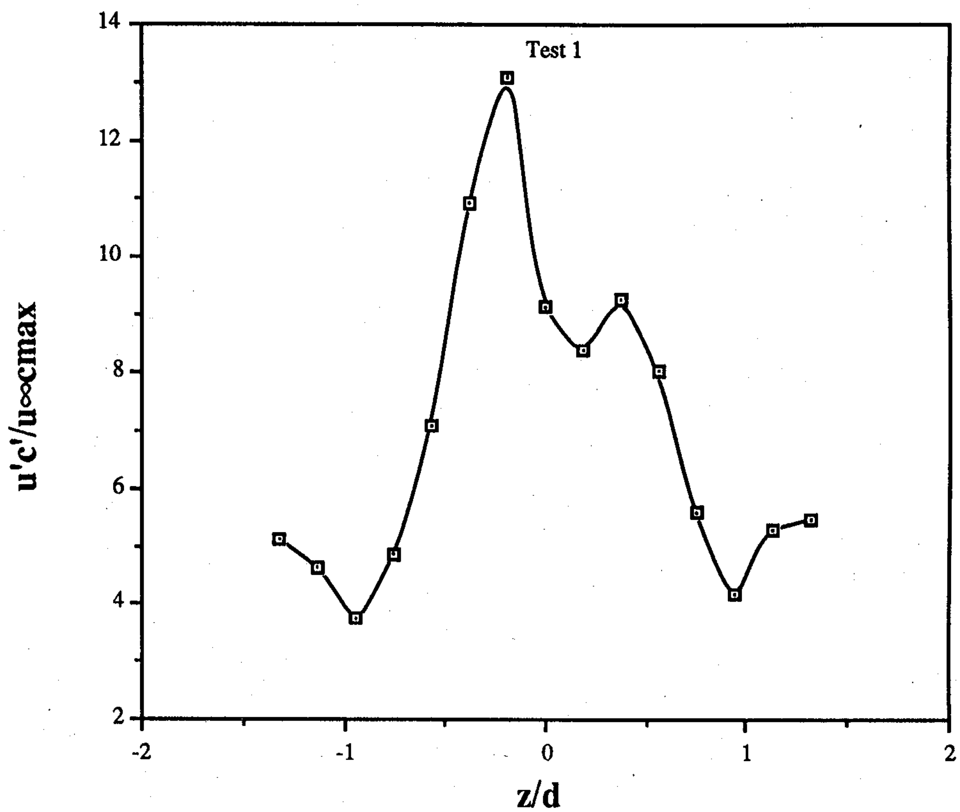



Fig. 16 Velocity-Concentration
Cross-Correlation

## **Appendix A.** The Traverse Control System

The traverse control system is made up of four subsystems, see Fig. A1. The first subsystem is the main data taking computer, the HP 9000-330. The second subsystem, the TCS8 (Traverse Control System 8 Axis), receives high level traverse commands from the HP 9000. The full duplex serial communications that links these two subsystems allows the HP 9000 to monitor the position and status of each axis in the system, see Appendix A.3 TCS8's Serial Interface Command Descriptions. The TCS8 can also function as a stand alone traverse controller. Through the use of the TCS8's front panel, an operator can execute all of the commands that the HP 9000 can, plus the operator can control all axes in jog mode, see Appendix A.1 TCS8's Front Panel Descriptions and Appendix A.2 TCS8's Local Command Descriptions. The third subsystem, the MDS (Motor Drive System), is controlled solely by the TCS8. The TCS8 translates the high level commands from the HP 9000 and its front panel into low level indexer commands, see The Compumotor AX Drive User Manual previously delivered. The TCS8 also receives encoder pulses from the traverses via the MDS. This allows the TCS8 to display realtime position information on its front panel. The fourth and final subsystem of the traverse control system is the slide, motor, encoder, and limit switches that make up each axis. A drawing of each cable that is used to connect the traverse control system is included in Appendix A.4 Traverse Control System Cables.

### The TCS8

The TCS8 is a microprocessor controlled system designed to interface an operator with a traverse system. The operator can utilize the TCS8 through the front panel, see Appendix A.1 TCS8's Front Panel Descriptions and Appendix A.2 TCS8's Local Command Descriptions, and/or with one or two host computers over serial interfaces, see Appendix A.3 TCS8's Serial Interface Command Descriptions. The TCS8 stores all the critical parameters of motion, for each of the eight axes that it controls, in non-volatile memory. The critical parameters of motion being: position, encoder counts per unit travel, encoder counts per motor revolution, velocity, and acceleration. All of these parameters may be viewed, set, and saved. The TCS8 has three modes of motion. They are absolute, relative, and jogged. With absolute movements, the operator specifies the final location; with relative movements, a distance is specified; and with jogged movements the operator presses a jog key on the front panel of the TCS8 until the desired location is obtained.

### The Motor Drive System

Each of the two MDS's have 4 indexer/drivers contained within them. The TCS8 communicates with the indexers in the MDS's over a closed loop serial daisy chain. When two MDS's are used, as in this setup, the first MDS in the chain must be set to 8 and the second set to 4. By setting the first MDS to 8, the operator is opening the closed loop serial daisy chain allowing the second MDS to be included in the chain. The 4/8 switch is located on the back panel of each MDS, see Fig. A2. This figure also shows the location of all the motor, limit, and encoder

connections. Channels X1, X2, Y1, and Y2 of the TCS8 control axis 1 through 4 on the first MDS and channels Z1, Z2, A1, and A2 control axis 1 through 4 on the second MDS. The TCS8 Encoders connector on the back of each MDS has a corresponding connector of the back of the TCS8, see Fig. A3 Schematic of TCS8 Back Panel. The interconnecting cable is detailed in Appendix A.4 Traverse Control System Cables.

**Positioning Resolution**

The indexer/drivers that are used in the MDS can drive the motors at 12,800 steps/revolution. The encoder used on each axis are 1000 pulses/revolution with quadrature encoding. Quadrature encoding adds a factor of 4 to the number of pulses/revolution to make this number 4000 pulses/revolution. This number, 4000 pulses/revolution, is well within the limit of 12;800 steps/revolution set by the indexer. The final factor in the product of the resolution of an axis is the number of threads/inch of the lead screw. All of the axes of the traverse system, except the auxiliary axis, have lead screws of 5 threads/inch, the auxiliary axis has a lead screw of 10 threads/inch. So the positioning resolution of the axes with a 5 threads/inch lead screw is 0.00005 inches and the auxiliary axis has a resolution of 0.000025 inches.

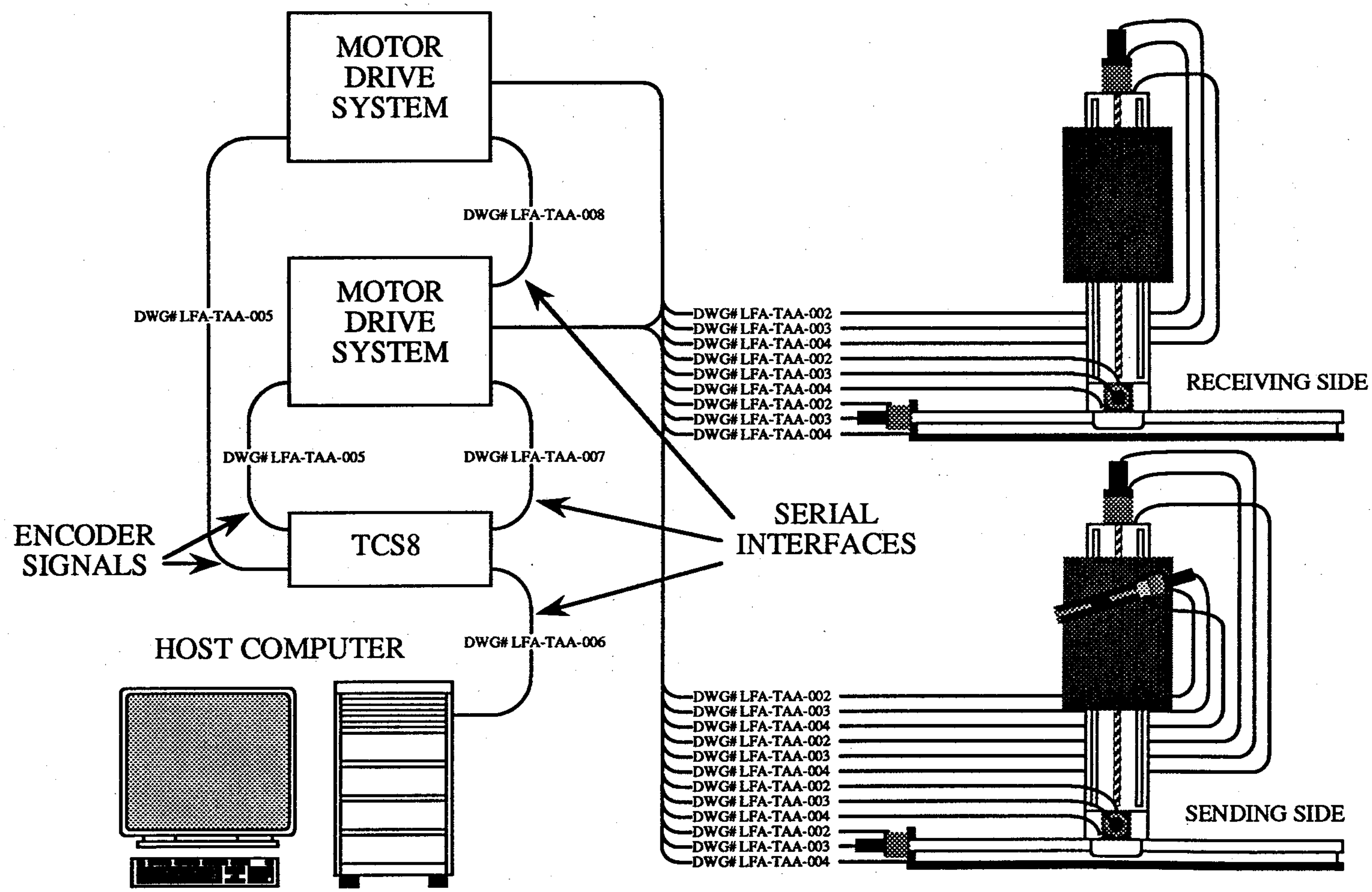



Fig. A1 Langley Traverse Control System

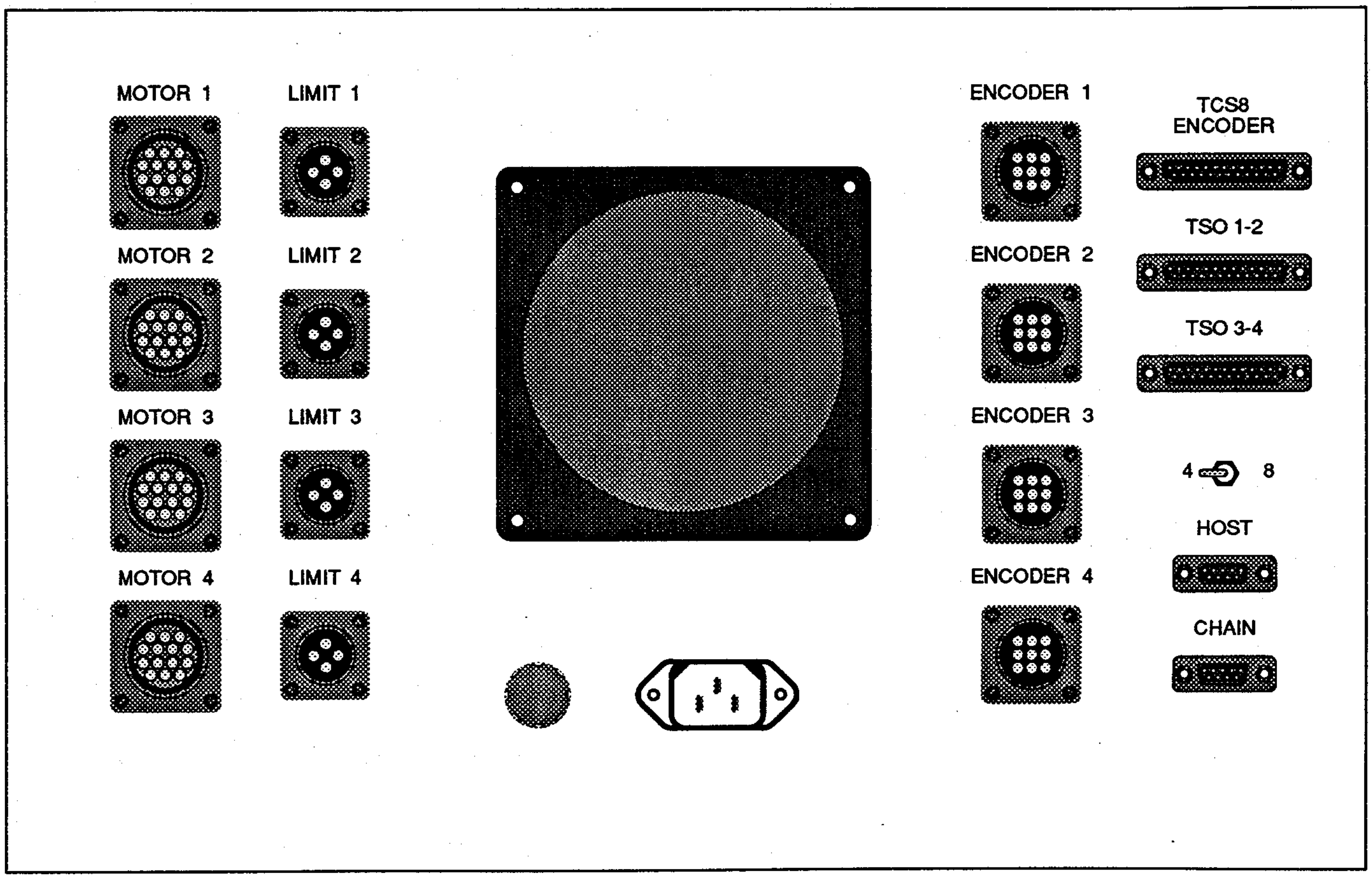



Fig. A2 Schematic of Motor Drive System Back Panel

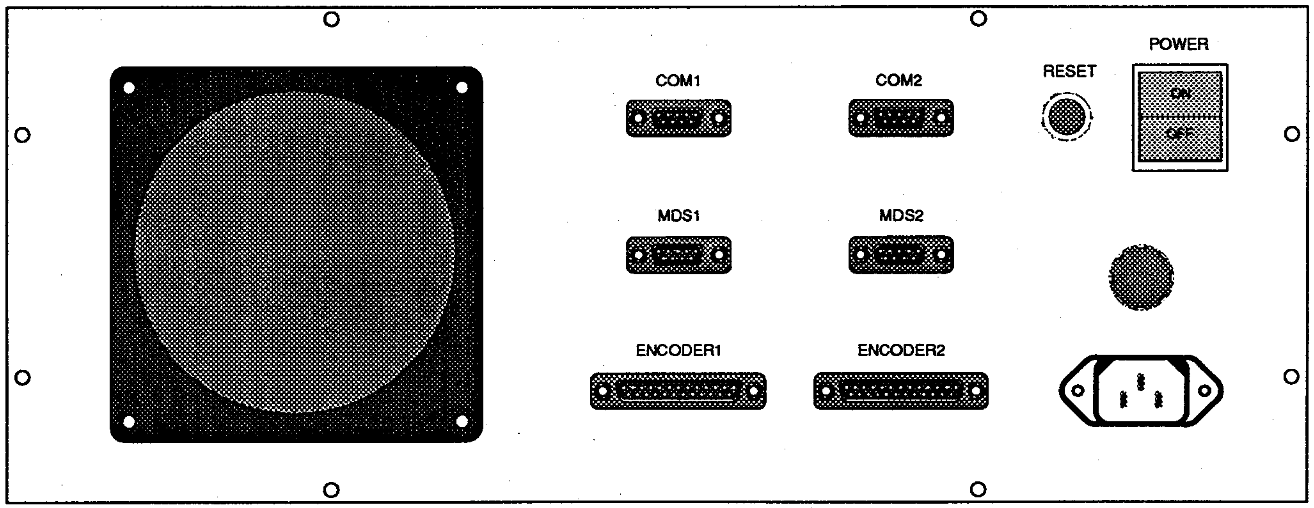
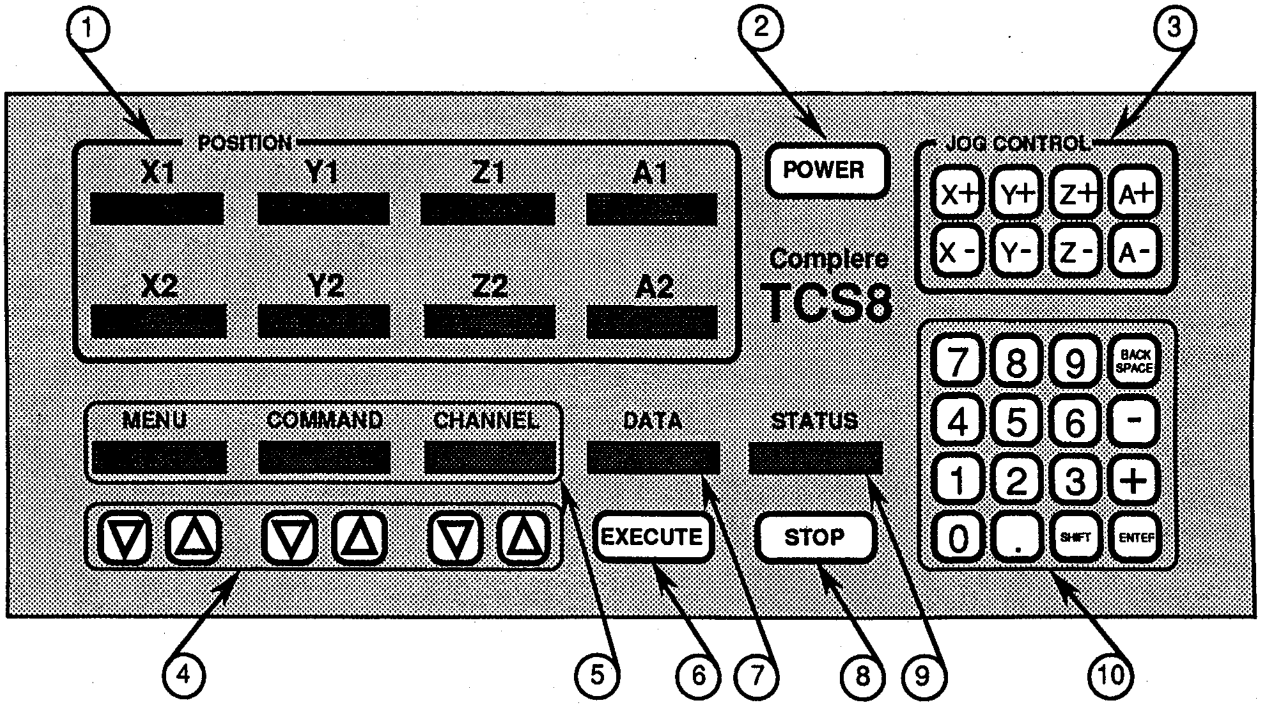



Fig. A3 Schematic of TCS8 Back Panel

## Appendix A.1 TCS8's Front Panel Description




### 1. POSITION DISPLAY WINDOWS

There are eight windows corresponding to the eight axes that the TCS8 is capable of controlling. The position of each axis is continuously updated, by monitoring its encoder, and displayed in a fixed format of a sign, two digits, a decimal point, and four digits.

### 2. POWER KEY

The power key is used to store the current configuration to nonvolatile memory before turning off power to the TCS8. Pressing the power key turns the displays off and saves the current configuration. Pressing it again turns the displays back on. This key can be used to implement a screen saver function.

### 3. JOG CONTROL KEYS

These keys are used to control up to eight axes in a jog mode. The mode (slaved, one's only, or two's only) can be set through the jog menu. When the operator presses a jog key, the respective axis will begin to move. The direction that the axis moves is determined by the operator pressing either a plus or minus jog key. A plus jog key will turn the lead screw in a clockwise direction (away from the motor), a minus jog key will turn it in the counter-clockwise direction (towards the motor). By releasing the jog key the operator stops motion on that axis. Motion will also stop, if the axis reaches the limit for the direction it is moving, or if the indexer determines that the axis has stalled.

## 4. SCROLL KEYS

These keys are used to scroll items through the MENU, COMMAND, and CHANNEL windows. All of the menus, their commands, and channel variations will be detailed in another appendix.

## 5. COMMAND WINDOWS

These three windows (MENU, COMMAND, and CHANNEL) are used, in tandem with their respective scroll keys, to formulate a command to be executed by the TCS8.

## 6. EXECUTE KEY

This key is used to execute the command currently formulated in the MENU, COMMAND, and CHANNEL windows.

## 7. DATA WINDOW

Many of the TCS8's commands require some added data, e.g. the distance to move or an axis' encoder counts per unit. Data for these commands is entered from the numeric key pad on the lower right of the TCS8 into the DATA window. Only a valid real number can be entered into the DATA window. If the operator enters an invalid real number the character that is invalid will flash until the operator presses backspace or a valid character.

## 8. STOP KEY

The stop key, when pressed, will stop motion on all axes. The TCS8 will not loose track of the position of any axis. A move command started by the host computer and stopped by the stop key will finish normally with the position being reported. The position reported is the instantaneous position when the stop key was pressed. The final position of the axis being moved could be different than what was reported thus the host computer should read the position again after a panic stop.

## 9. STATUS WINDOW

The STATUS window reflects the result of all commands. For commands that are not instantaneous, this window displays a busy status and then when the command completes it displays a ready status. The results of all view commands are displayed in the STATUS window. The STATUS window also displays the activity over the COM interfaces. For example, when the command for viewing position is sent over the COM1 interface, the STATUS window will display "COM1 VP" and when the command is completed the window will display "COM1 vp".

## 10. NUMERIC KEY PAD

The numeric key pad is used to enter a number into the data window. The user may backspace in the window or clear (shift-backspace) the window. Only a valid real number can be entered into the data window. If the operator enters an invalid real number the character that is invalid will flash until the operator presses backspace or a valid character.

## Appendix A.2 TCS8's Local Command Descriptions

This appendix describes the command set that can be executed from the front panel of the TCS8. Using the up and down keys under the MENU, COMMAND, and CHANNEL windows, the operator can formulate a command and then execute it by pressing the EXECUTE key. Some commands require extra information to be entered into the DATA window through the use of the numeric key pad. Each description includes a list of related commands that should be refer to to enhance the operator's understanding of the command. Also where applicable, the default setting is given.

**MENU:** MOVE

**COMMAND:** TO ZERO

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The MOVE TO ZERO command is an easy way to move some or all of the axes to the zero position. This command can also be accomplished with the MOVE ABSOLUTE command and a zero in the DATA window. Before using this command, selected axes must be initialized with the INIT DRIVE ON command. This command can be canceled by pressing the STOP key. When the STOP key is pressed, all axes will stop immediately. If an axis encounters a limit before reaching zero, the rest of its movement is aborted.

**RELATED COMMANDS:** MOVE ABSOLUTE, MOVE RELATIVE, INIT Drive ON

---

**MENU:** MOVE

**COMMAND:** ABSOLUTE

**CHANNELS:** X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The MOVE ABSOLUTE command requires a position to be entered in the DATA window. This position and the current position of the axis is used to calculate the relative distance the axis must move. Before using this command, selected axes must be initialized with the INIT DRIVE ON command. This command can be canceled by pressing the STOP key. When the STOP key is pressed, all axes will stop immediately. If an axis encounters a limit before reaching the position entered in the DATA window, the rest of its movement is aborted.

**RELATED COMMANDS:** MOVE TO ZERO, MOVE RELATIVE, INIT Drive ON

**MENU:** MOVE

**COMMAND:** RELATIVE

**CHANNELS:** X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The MOVE RELATIVE command requires a distance to be entered in the DATA window. This position is used to calculate the relative distance the axis must move. Before using this command, selected axes must be initialized with the INIT DRIVE ON command. This command can be canceled by pressing the STOP key. When the STOP key is pressed, all axes will stop immediately. If an axis encounters a limit before moving the distance entered in the DATA window, the rest of its movement is aborted.

**RELATED COMMANDS:** MOVE TO ZERO, MOVE ABSOLUTE, INIT Drive ON

---

**MENU:** JOG

**COMMAND:** MODE

**CHANNELS:** SLAVED, ONE'S, TWO'S

**DESCRIPTION:** The JOG MODE command sets the way the JOG keys operate. When SLAVED is the setting, both the one and two axis of the X, Y, Z, or A coordinate will move the same amount. When ONE'S is the setting, only the one axes of the X, Y, Z, or A coordinate will move. And finally, when TWO'S is the setting, only the two axes of the X, Y, Z, or A coordinate will move. The current mode is marked with an asterisk.

**RELATED COMMANDS:** none

**DEFAULT:** SLAVED

**MENU:** SET

**COMMAND:** CPU

**CHANNELS:** ALL, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET CPU command allows the user to change the counts per unit travel. The CPU for an axis is determined by multiplying the encoder resolution (counts/revolution) by the lead screws resolution (revolutions/unit of travel). A units conversion can be added here to change for example from inches to centimeters. When the CPU for an axis is changed, the position is automatically converted. This command requires a value to be entered in the DATA window.

**RELATED COMMANDS:** SET CPR, SET POSITION

**DEFAULT:**

| | |
|---|---|
| X1 | 20000 |
| X2 | 20000 |
| Y1 | 20000 |
| Y2 | 20000 |
| Z1 | 20000 |
| Z2 | 20000 |
| A1 | 40000 |
| A2 | 40000 |

---

**MENU:** SET

**COMMAND:** CPR

**CHANNELS:** ALL, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET CPR command allows the user to change the encoder counts per motor revolution. The CPR for an axis is determined by dividing the encoder resolution (counts/revolution) by the lead screws resolution (revolutions/unit of travel). The encoder counts per motor revolution, that is entered in the DATA window, must be a positive integer.

**RELATED COMMANDS:** SET CPU

**DEFAULT:**

| | |
|---|---|
| X1 | 4000 |
| X2 | 4000 |
| Y1 | 4000 |
| Y2 | 4000 |
| Z1 | 4000 |
| Z2 | 4000 |
| A1 | 4000 |
| A2 | 4000 |

**MENU:** SET

**COMMAND:** POSITION

**CHANNELS:** ALL, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET POSITION command allows the user to change the current position of an axis. The new position must be entered in the DATA window be for executing the command.

**RELATED COMMANDS:** SET CPU

---

**MENU:** SET

**COMMAND:** VELOCITY

**CHANNELS:** ALL, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET VELOCITY command allows the user to change the maximum speed at which an axis will travel. The range of valid velocities is 0.002 to 50.000 revolutions per second. The default is 5 revs/sec. An axis may stall at velocities higher than the default. The new velocity must be entered in the DATA window be for executing the command.

**RELATED COMMANDS:** SET ACCEL.

| | | |
|---|---|---|
| **DEFAULT:** | X1 | 5.000 |
| | X2 | 5.000 |
| | Y1 | 5.000 |
| | Y2 | 5.000 |
| | Z1 | 5.000 |
| | Z2 | 5.000 |
| | A1 | 5.000 |
| | A2 | 5.000 |

**MENU:** SET

**COMMAND:** ACCEL.

**CHANNELS:** ALL, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET ACCEL. command allows the user to change the maximum acceleration for an axis. The range of valid accelerations is 0.01 to 999.99 revolutions per second per second. The default is 5 revs/sec/sec. The new acceleration must be entered in the DATA window be for executing the command.

**RELATED COMMANDS:** SET VELOCITY

**DEFAULT:**

| | |
|---|---|
| X1 | 5.00 |
| X2 | 5.00 |
| Y1 | 5.00 |
| Y2 | 5.00 |
| Z1 | 5.00 |
| Z2 | 5.00 |
| A1 | 5.00 |
| A2 | 5.00 |

---

**MENU:** SET

**COMMAND:** CrntsOn

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET CrntsOn command allows the user to turn the motor currents on. The motor current must be on for an axis to be moved. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CrntsOff

**MENU:** SET

**COMMAND:** CrntsOff

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET CrntsOff command allows the user to power down motors when they will not be used for long periods of time. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CrntsOn

---

**MENU:** SET

**COMMAND:** INITS ON

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The SET INITS ON command allows the user to initialize the indexers without turning on the power to the motors. The information in the DATA window is ignored.

**RELATED COMMANDS:** INIT Drive ON

---

**MENU:** VIEW

**COMMAND:** Cnt/Unit

**CHANNELS:** X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The VIEW Cnt/Unit command displays the current setting of the encoder counts per unit travel parameter for the selected axis in the STATUS window. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CPU

**MENU:** VIEW

**COMMAND:** Cnt/MRev

**CHANNELS:** X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The VIEW Cnt/MRev command displays the current setting of the encoder counts per motor revolution parameter for the selected axis in the STATUS window. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CPR

---

**MENU:** VIEW

**COMMAND:** VELOCITY

**CHANNELS:** X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The VIEW VELOCITY command displays the current setting of the velocity parameter for the selected axis in the STATUS window. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET VELOCITY

---

**MENU:** VIEW

**COMMAND:** ACCEL.

**CHANNELS:** X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The VIEW ACCEL. command displays the current setting of the acceleration parameter for the selected axis in the STATUS window. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET ACCEL.

**MENU:** VIEW

**COMMAND:** INIT

**CHANNELS:** none

**DESCRIPTION:** The VIEW INIT command uses the STATUS window to display a one(initialized) or a zero(uninitialized) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET INITS, INIT Drive ON

---

**MENU:** VIEW

**COMMAND:** CURRENTS

**CHANNELS:** none

**DESCRIPTION:** The VIEW CURRENTS command uses the STATUS window to display a one(current on) or a zero(current off) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CrntsOn, SET CrntsOff, INIT Drive ON, INIT Drive OFF

---

**MENU:** VIEW

**COMMAND:** Plus LMT

**CHANNELS:** none

**DESCRIPTION:** The VIEW Plus LMT command uses the STATUS window to display a one(on limit) or a zero(not on limit) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** none

**MENU:** VIEW

**COMMAND:** Minus LMT

**CHANNELS:** none

**DESCRIPTION:** The VIEW Minus LMT command uses the STATUS window to display a one(on limit) or a zero(not on limit) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** none

---

**MENU:** VIEW

**COMMAND:** HOME

**CHANNELS:** none

**DESCRIPTION:** The VIEW HOME command uses the STATUS window to display a one(on limit) or a zero(not on limit) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** none

---

**MENU:** VIEW

**COMMAND:** STALL

**CHANNELS:** none

**DESCRIPTION:** The VIEW STALL command uses the STATUS window to display a one(stalled) or a zero(not stalled) for each axis. The STATUS window has eight characters; left to right respectively reflecting the status of: X1, X2 ..., A1, A2. The information in the DATA window is ignored.

**RELATED COMMANDS:** none

**MENU:** INIT

**COMMAND:** DEFAULT

**CHANNELS:** none

**DESCRIPTION:** The INIT DEFAULT command restores the initial factory defaults (CPU, CPR, VELOCITY, ACCELERATION,BAUD RATE,BITS/CHAR,PARITY,STOP BITS,HANDSHAKE) of the TCS8. After executing this command, execute the command INIT Drive ON to initialize the indexers. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CPU, SET CPR, SET VELOCITY, SET ACCEL.

---

**MENU:** INIT

**COMMAND:** Drive ON

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The INIT Drive ON command initializes the selected axes for movement. After executing this command the currents are on to the motors. The information in the DATA window is ignored.

**RELATED COMMANDS:** SET CPU, SET CPR, SET VELOCITY, SET ACCEL., SET CnrtsOn, SET CnrtsOff, INIT DEFAULT

---

**MENU:** INIT

**COMMAND:** Drive OFF

**CHANNELS:** ALL, ONE'S, TWO'S, X1&X2, Y1&Y2, Z1&Z2, A1&A2, X1, X2, Y1, Y2, Z1, Z2, A1, A2

**DESCRIPTION:** The INIT Drive OFF command is an alias for SET CrntsOff.

**RELATED COMMANDS:** SET CrntsOff

**MENU:** COM1/COM2

**COMMAND:** BaudRate

**CHANNELS:** 19.2K, 9600, 4800, 2400, 1200, 300, 110

**DESCRIPTION:** The COM1/COM2 BaudRate command set the baud rate for the selected communication channel. The information in the DATA window is ignored. The current baud rate is marked with an asterisk.

**RELATED COMMANDS:** none

**DEFAULT:** 9600

---

**MENU:** COM1/COM2

**COMMAND:** Bit/Char

**CHANNELS:** SEVEN, EIGHT

**DESCRIPTION:** The COM1/COM2 Bit/Char command set the bits per character for the selected communication channel. The information in the DATA window is ignored. The current number of bits per character is marked with an asterisk.

**RELATED COMMANDS:** none

**DEFAULT:** EIGHT

---

**MENU:** COM1/COM2

**COMMAND:** Parity

**CHANNELS:** NONE, EVEN, ODD

**DESCRIPTION:** The COM1/COM2 Parity command set the parity for the selected communication channel. The information in the DATA window is ignored. The current parity is marked with an asterisk.

**RELATED COMMANDS:** none

**DEFAULT:** EVEN

**MENU:** COM1/COM2

**COMMAND:** StopBits

**CHANNELS:** 1, 1.5, 2

**DESCRIPTION:** The COM1/COM2 StopBits command set the stop bits for the selected communication channel. The information in the DATA window is ignored. The current number of stop bits is marked with an asterisk.

**RELATED COMMANDS:** none

**DEFAULT:** 1

---

**MENU:** COM1/COM2

**COMMAND:** HandShak

**CHANNELS:** NO, YES

**DESCRIPTION:** The COM1/COM2 HandShak command set the handshake for the selected communication channel. The information in the DATA window is ignored. An asterisk marks whether there is handshaking or not.

**RELATED COMMANDS:** none

**DEFAULT:** YES

## Appendix A.3 TCS8's Serial Interface Command Descriptions

This appendix describes the command set that can be executed through the serial interfaces of the TCS8. Each description includes a code section that outlines the characters that must be sent to execute the command. The vertical bar in this section is used as a separator and is not sent as part of the command code. The symbol "CRLF" stands for the two characters carriage return and line feed. Also where applicable, the default setting is given.

**COMMAND:** CHANGE SERIAL CONFIGURATION

**CODE:** CS COM;CATEGORY;ATTRIBUTE;

| | | |
|---|---|---|
| **PARAMETERS:** | COM: | 1/COM1 |
| | | 2/COM2 |
| | CATEGORY: | 0/BAUDRATE |
| | ATTRIBUTE: | 0/19.2K |
| | | 1/9600 |
| | | 2/4800 |
| | | 3/2400 |
| | | 4/1200 |
| | | 5/300 |
| | | 6/110 |
| | CATEGORY: | 1(BITS PER CHARACTER) |
| | ATTRIBUTE: | 0/SEVEN |
| | | 1/EIGHT |
| | CATEGORY: | 2(PARITY) |
| | ATTRIBUTE: | 0/NONE |
| | | 1/EVEN |
| | | 2/ODD |
| | CATEGORY: | 3(STOP BITS) |
| | ATTRIBUTE: | 0/ONE |
| | | 1/ONE AND A HALF |
| | | 2/TWO |
| | CATEGORY: | 4(HANDSHAKE) |
| | ATTRIBUTE: | 0/NO |
| | | 1/YES |

**DESCRIPTION:** This command must be executed with extreme caution and thought. If the user changes an attribute of the same COM port that he is sending the command, he must change to that attribute on the host computer before sending the next command. The best way to change the serial configuration of a COM port is to utilize the front panel commands.

**DEFAULT:** 9600 baud, EIGHT bits/char, EVEN parity, ONE stop bit, handshaking YES

**EXAMPLE:** To change the baudrate of COM1 to 2400 the user must send CS1;0;3;

**COMMAND:** MOVE TO ABSOLUTE POSITION AND REPORT FINAL POSITION

**CODE:** MA CHANNEL:POSITION,CHANNEL:POSITION,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS |
| | | 1/X1 |
| | | 2/X2 |
| | | 3/Y1 |
| | | 4/Y2 |
| | | 5/Z1 |
| | | 6/Z2 |
| | | 7/A1 |
| | | 8/A2 |
| | POSITION: | Real number free format |

**DESCRIPTION:** This command moves selected channels to absolute positions.

**EXAMPLES:**
To move all channels to zero the user may send MA0:0,CRLF or MA12345678:0,CRLF
To move channel X1 to zero the user must send MA1:0,CRLF
To move channels X1 and X2 to zero the user may send MA12:0,CRLF or MA1:0,2:0,CRLF or MA1:0,CRLF and MA2:0,CRLF

---

**COMMAND:** MOVE TO RELATIVE DISTANCE AND REPORT FINAL POSITION

**CODE:** MR CHANNEL:DISTANCE,CHANNEL:DISTANCE,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS |
| | | 1/X1 |
| | | 2/X2 |
| | | 3/Y1 |
| | | 4/Y2 |
| | | 5/Z1 |
| | | 6/Z2 |
| | | 7/A1 |
| | | 8/A2 |
| | POSITION: | Real number free format |

**DESCRIPTION:** This command moves selected channels relative distances.

**EXAMPLES:**
To move all channels one unit the user may send MR0:1,CRLF or MR12345678:1,CRLF
To move channel X1 one unit the user must send MR1:1,CRLF
To move channels X1 and X2 one unit the user may send MR12:1,CRLF or MR1:1,2:1,CRLF or MR1:1,CRLF and MR2:1,CRLF

**COMMAND:** SET ACCELERATION

**CODE:** SA CHANNEL:ACCELERATION,CHANNEL:ACCELERATION,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |
| | ACCELERATION: | Real number free format between<br>0.01 and 99.99 inclusive. |

**DESCRIPTION:** This command sets the acceleration for selected channels.

**DEFAULT:** All channels 5.00 revolutions/second/second

**EXAMPLES:**
To set the acceleration for all channels to 4.00 revolutions/second/second the user may send SA0:4.00,CRLF or SA12345678:4.00,CRLF
To set the acceleration for channel X1 to 4.00 revolutions/second/second the user must send SA1:4.00,CRLF
To set the acceleration for channels X1 and X2 to 4.00 revolutions/second/second the user may send SA12:4.00,CRLF or SA1:4.00 ,2:4.00,CRLF or SA1:4.00,CRLF and SA2:4.00,CRLF

---

**COMMAND:** VIEW ACCELERATION

**CODE:** VA CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |

**DESCRIPTION:** This command views the acceleration for selected channels. The TCS8 transmits each of the accelerations requested back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the acceleration for all channels the user may send VA0CRLF or VA12345678CRLF
To view the acceleration for channel X1 the user must send VA1CRLF
To view the acceleration for channels X1 and X2 the user may send VA12CRLF or VA1CRLF and VA2CRLF

**COMMAND:** SET VELOCITY

**CODE:** SV CHANNEL:VELOCITY,CHANNEL:VELOCITY,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |
| | VELOCITY: | Real number free format between<br>0.001 and 50.000 inclusive. |

**DESCRIPTION:** This command sets the velocity for selected channels.

**DEFAULT:** All channels 5.000 revolutions/second

**EXAMPLES:**
To set the velocity for all channels to 4.00 revolutions/second the user may send SV0:4.00,CRLF or SV12345678:4.00,CRLF
To set the velocity for channel X1 to 4.00 revolutions/second the user must send SV1:4.00,CRLF
To set the velocity for channels X1 and X2 to 4.00 revolutions/second the user may send SV12:4.00,CRLF or SV1:4.00 ,2:4.00,CRLF or SV1:4.00,CRLF and SV2:4.00,CRLF

---

**COMMAND:** VIEW VELOCITY

**CODE:** VV CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |

**DESCRIPTION:** This command views the velocity for selected channels. The TCS8 transmits each of the velocities requested back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the velocity for all channels the user may send VV0CRLF or VV12345678CRLF
To view the velocity for channel X1 the user must send VV1CRLF
To view the velocity for channels X1 and X2 the user may send VV12CRLF or VV1CRLF and VV2CRLF

**COMMAND:** SET ENCODER COUNTS PER UNIT TRAVEL

**CODE:** SU CHANNEL:CPU,CHANNEL:CPU,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |
| | CPU: | Non-zero real number free format. |

**DESCRIPTION:** This command sets the encoder counts per unit travel for selected channels.

**DEFAULT:** X1,2,Y1,Y2,Z1,Z2 20000 counts/inch and A1,A2 40000 counts/inch

**EXAMPLES:**
To set the encoder counts per unit travel for all channels to 5000 the user may send SU0:5000,CRLF or SU12345678:5000,CRLF
To set the encoder counts per unit travel for channel X1 to 5000 the user must send SU1:5000,CRLF
To set the encoder counts per unit travel for channels X1 and X2 to 5000 the user may send SU12:5000,CRLF or SU1:5000 ,2:5000,CRLF or SU1:5000,CRLF and SU2:5000,CRLF

---

**COMMAND:** VIEW ENCODER COUNTS PER UNIT TRAVEL

**CODE:** VU CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |

**DESCRIPTION:** This command views the encoder counts per unit travel for selected channels. The TCS8 transmits each of the encoder counts per unit travel requested back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the encoder counts per unit travel for all channels the user may send VU0CRLF or VU12345678CRLF
To view the encoder counts per unit travel for channel X1 the user must send VU1CRLF
To view the encoder counts per unit travel for channels X1 and X2 the user may send VU12CRLF or VU1CRLF and VU2CRLF

**COMMAND:** SET ENCODER COUNTS PER MOTOR REVOLUTION

**CODE:** SR CHANNEL:CPR,CHANNEL:CPR,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS |
| | | 1/X1 |
| | | 2/X2 |
| | | 3/Y1 |
| | | 4/Y2 |
| | | 5/Z1 |
| | | 6/Z2 |
| | | 7/A1 |
| | | 8/A2 |
| | CPU: | Non-zero integer free format. |

**DESCRIPTION:** This command sets the encoder counts per motor revolution for selected channels.

**DEFAULT:** X1,2,Y1,Y2,Z1,Z2 and A1,A2 4000 counts/inch

**EXAMPLES:**
To set the encoder counts per motor revolution for all channels to 500 the user may send SR0:500,CRLF or SR12345678:500,CRLF
To set the encoder counts per motor revolution for channel X1 to 500 the user must send SR1:500,CRLF
To set the encoder counts per motor revolution for channels X1 and X2 to 500 the user may send SR12:500,CRLF or SR1:500 ,2:500,CRLF or SR1:500,CRLF and SR2:500,CRLF

---

**COMMAND:** VIEW ENCODER COUNTS PER MOTOR REVOLUTION

**CODE:** VR CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS |
| | | 1/X1 |
| | | 2/X2 |
| | | 3/Y1 |
| | | 4/Y2 |
| | | 5/Z1 |
| | | 6/Z2 |
| | | 7/A1 |
| | | 8/A2 |

**DESCRIPTION:** This command views the encoder counts per motor revolution for selected channels. The TCS8 transmits each of the encoder counts per motor revolution requested back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the encoder counts per motor revolution for all channels the user may send VR0CRLF or VR12345678CRLF
To view the encoder counts per motor revolution for channel X1 the user must send VR1CRLF
To view the encoder counts per motor revolution for channels X1 and X2 the user may send VR12CRLF or VR1CRLF and VR2CRLF

**COMMAND:** SET POSITION

**CODE:** SP CHANNEL:POSITION,CHANNEL:POSITION,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |
| | POSITION: | real number. |

**DESCRIPTION:** This command sets the position for selected channels.

**EXAMPLES:**
To set the position for all channels to 1.5 the user may send SP0:1.5,CRLF or SP12345678:1.5,CRLF
To set the position for channel X1 to 1.5 the user must send SP1:1.5,CRLF
To set the position for channels X1 and X2 to 1.5 the user may send SP12:1.5,CRLF or SP1:1.5 ,2:1.5,CRLF or SP1:1.5,CRLF and SP2:1.5,CRLF

---

**COMMAND:** VIEW POSITION

**CODE:** VP CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |

**DESCRIPTION:** This command views the position for selected channels. The TCS8 transmits each of the positions requested back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the position for all channels the user may send VP0CRLF or VP12345678CRLF
To view the position for channel X1 the user must send VP1CRLF
To view the position for channels X1 and X2 the user may send VP12CRLF or VP1CRLF and VP2CRLF

**COMMAND:** SET CURRENT TO MOTOR WINDINGS

**CODE:** SC CHANNEL:ON/OFF,CHANNEL:ON/OFF,...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |
| | ON/OFF: | 1/ON<br>0/OFF |

**DESCRIPTION:** This command sets the current to the motor windings for selected channels on or off.

**EXAMPLES:**
To set the current to the motor windings for all channels on the user may send SC0:1,CRLF or SC12345678:1,CRLF to set them off the user may send SC0:0,CRLF or SC12345678:0,CRLF

---

**COMMAND:** VIEW CURRENT TO MOTOR WINDINGS

**CODE:** VC CHANNEL|CHANNEL...|CRLF

| | | |
|---|---|---|
| **PARAMETERS:** | CHANNEL: | 0/ALL CHANNELS<br>1/X1<br>2/X2<br>3/Y1<br>4/Y2<br>5/Z1<br>6/Z2<br>7/A1<br>8/A2 |

**DESCRIPTION:** This command views the current to the motor windings for selected channels. The TCS8 transmits each response of on/off (1/0) back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To view the current to the motor windings for all channels the user may send VC0CRLF or VC12345678CRLF
To view the current to the motor windings for channel X1 the user must send VC1CRLF
To view the current to the motor windings for channels X1 and X2 the user may send VC12CRLF or VC1CRLF and VC2CRLF

**COMMAND:** SET INITIALIZATION OF INDEXER/DRIVERS

**CODE:** SI CHANNEL|CHANNEL...|CRLF

**PARAMETERS:** CHANNEL:

0/ALL CHANNELS
1/X1
2/X2
3/Y1
4/Y2
5/Z1
6/Z2
7/A1
8/A2

**DESCRIPTION:** This command sends the current value of the acceleration, velocity, and the encoder counts per motor revolution to the indexer/driver for the selected channels. This command must be sent before any move commands may be sent.

**EXAMPLES:**
To initialize all channels the user may send SI0CRLF or SI12345678CRLF
To initialize channel X1 the user must send SI1CRLF
To initialize channels X1 and X2 the user may send SI12CRLF or SI1CRLF and SI2CRLF

---

**COMMAND:** VIEW INITIALIZATION OF INDEXER/DRIVERS

**CODE:** VI CHANNEL|CHANNEL...|CRLF

**PARAMETERS:** CHANNEL:

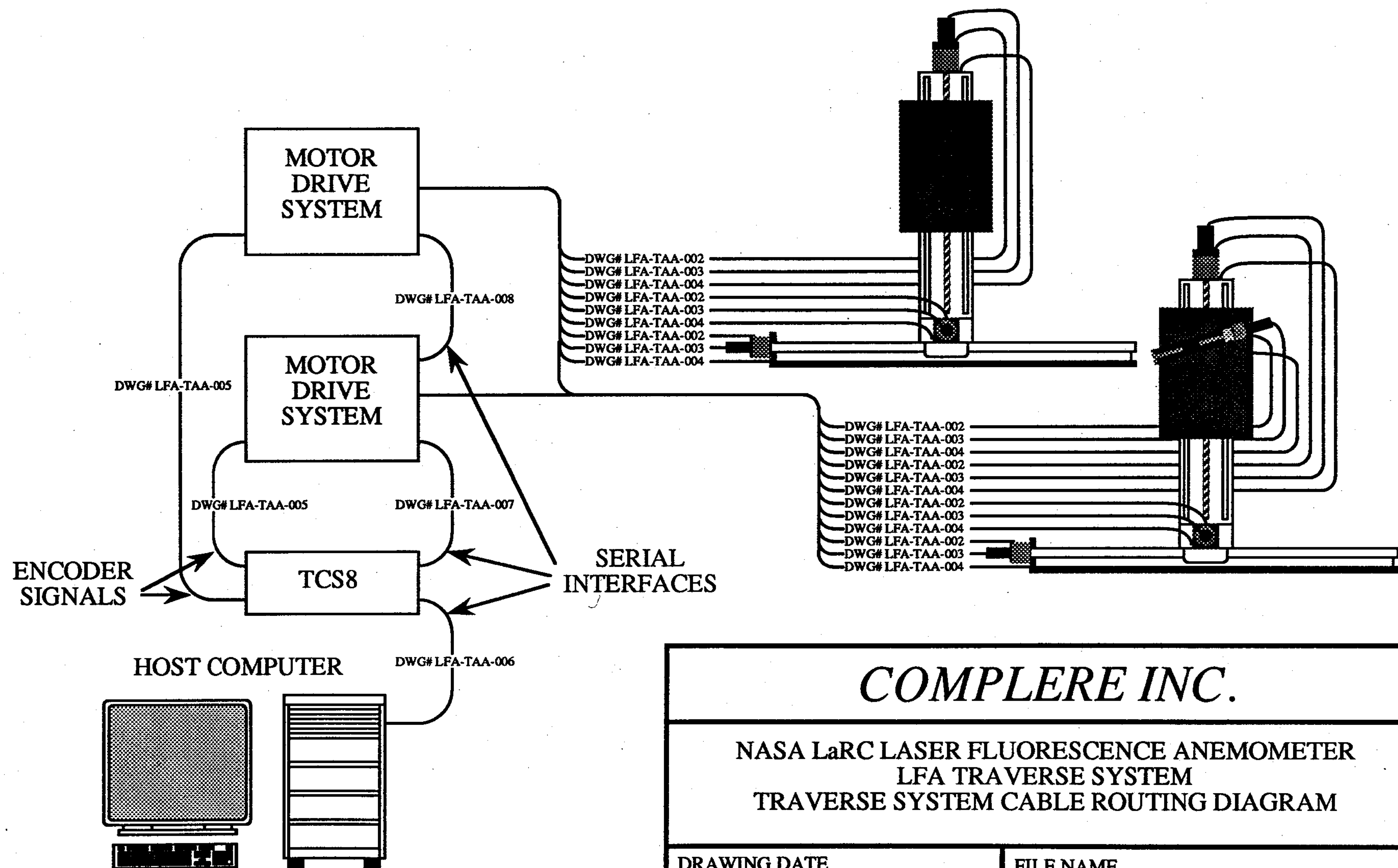
0/ALL CHANNELS
1/X1
2/X2
3/Y1
4/Y2
5/Z1
6/Z2
7/A1
8/A2

**DESCRIPTION:** This command returns "1" if the indexer/driver has been initialized since the TCS8 was turned on and "0" if it has not. The TCS8 transmits each of the responses back to the host computer separated by carriage return line feeds.

**EXAMPLES:**
To check the initialization of all channels the user may send VI0CRLF or VI12345678CRLF
To check the initialization of channel X1 the user must send VI1CRLF
To check the initialization of channels X1 and X2 the user may send VI12CRLF or VI1CRLF and VI2CRLF

# Appendix A.4

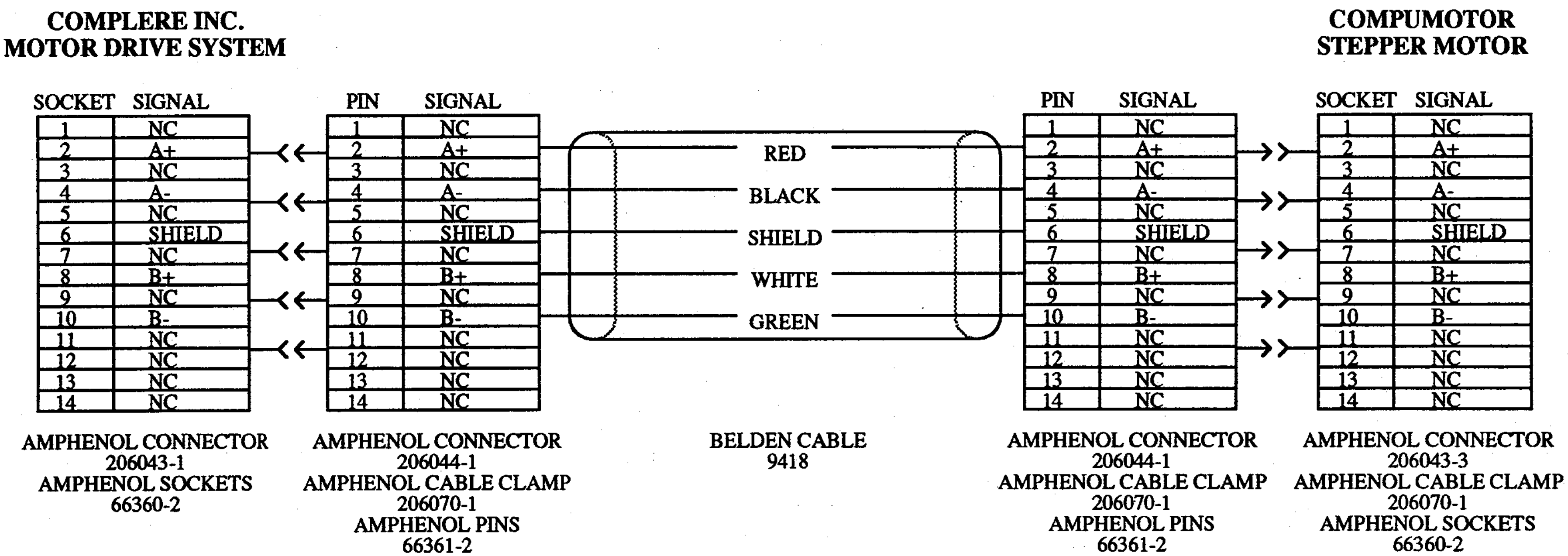
## Traverse Control System Cables




# *COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
TRAVERSE SYSTEM CABLE ROUTING DIAGRAM

| DRAWING DATE | FILE NAME |
|---|---|
| JULY 8, 1991 | LANGLEY TRAVERSE DIAGRAM |
| **DESIGN ENGINEER** | **DRAWING NUMBER** |
| TODD A. AMBUR | LFA-TAA-001 |




| SOCKET | SIGNAL | PIN | SIGNAL | PIN | SIGNAL | SOCKET | SIGNAL |
|---|---|---|---|---|---|---|---|
| 1 | NC | 1 | NC | 1 | NC | 1 | NC |
| 2 | A+ | 2 | A+ | 2 | A+ | 2 | A+ |
| 3 | NC | 3 | NC | 3 | NC | 3 | NC |
| 4 | A- | 4 | A- | 4 | A- | 4 | A- |
| 5 | NC | 5 | NC | 5 | NC | 5 | NC |
| 6 | SHIELD | 6 | SHIELD | 6 | SHIELD | 6 | SHIELD |
| 7 | NC | 7 | NC | 7 | NC | 7 | NC |
| 8 | B+ | 8 | B+ | 8 | B+ | 8 | B+ |
| 9 | NC | 9 | NC | 9 | NC | 9 | NC |
| 10 | B- | 10 | B- | 10 | B- | 10 | B- |
| 11 | NC | 11 | NC | 11 | NC | 11 | NC |
| 12 | NC | 12 | NC | 12 | NC | 12 | NC |
| 13 | NC | 13 | NC | 13 | NC | 13 | NC |
| 14 | NC | 14 | NC | 14 | NC | 14 | NC |

| SIGNAL | DESCRIPTION |
|---|---|
| A+ | Motor Winding |
| A- | Motor Winding |
| B+ | Motor Winding |
| B- | Motor Winding |
| SHIELD | Motor Case Ground |
| NC | No Connection |

*COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
MOTOR DRIVE SYSTEM TO COMPUMOTOR STEPPER MOTOR

| DRAWING DATE | FILE NAME |
|---|---|
| JULY 3, 1991 | LANGLEY MOTOR |
| **DESIGN ENGINEER** | **DRAWING NUMBER** |
| TODD A. AMBUR | LFA-TAA-002 |

**COMPLERE INC.**
**MOTOR DRIVE SYSTEM**

**DYNAMICS RESEARCH**
**ENCODER**

| PIN | SIGNAL | SOCKET | SIGNAL | | SOCKET | SIGNAL | PIN | SIGNAL |
|---|---|---|---|---|---|---|---|---|
| 1 | A+ | 1 | A+ | BLUE | 1 | A+ | 1 | A+ |
| 2 | A- | 2 | A- | BLACK | 2 | A- | 2 | A- |
| 3 | B+ | 3 | B+ | GREEN | 3 | B+ | 3 | B+ |
| 4 | B- | 4 | B- | BLACK | 4 | B- | 4 | B- |
| 5 | SHIELD | 5 | SHIELD | SHIELD | 5 | SHIELD | 5 | SHIELD |
| 6 | Z+ | 6 | Z+ | WHITE | 6 | Z+ | 6 | Z+ |
| 7 | Z- | 7 | Z- | BLACK | 7 | Z- | 7 | Z- |
| 8 | +5VDC | 8 | +5VDC | RED | 8 | +5VDC | 8 | +5VDC |
| 9 | GROUND | 9 | GROUND | BLACK | 9 | GROUND | 9 | GROUND |

AMPHENOL CONNECTOR
206705-1
AMPHENOL PINS
66103-2

AMPHENOL CONNECTOR
206708-1
AMPHENOL CABLE CLAMP
206966-1
AMPHENOL SOCKETS
66105-2

BELDEN CABLE
9504

AMPHENOL CONNECTOR
206708-1
AMPHENOL CABLE CLAMP
206966-1
AMPHENOL SOCKETS
66105-2

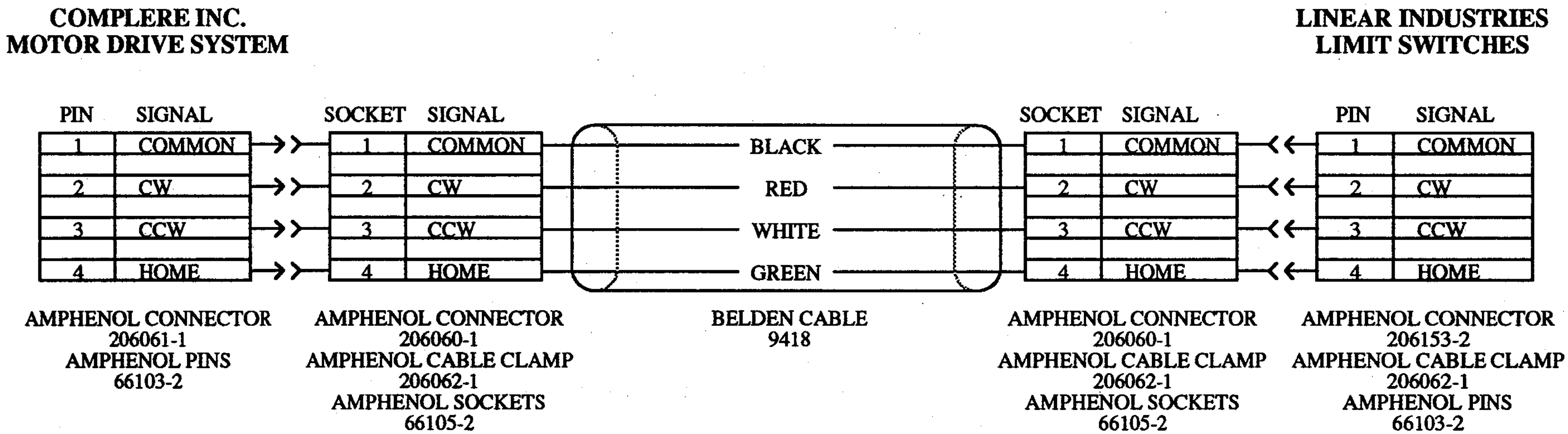
AMPHENOL CONNECTOR
206705-2
AMPHENOL CABLE CLAMP
206966-1
AMPHENOL PINS
66103-2

| SIGNAL | DESCRIPTION |
|---|---|
| A+ | Quadrature Encoder Signal |
| A- | Logical Complement of A+ |
| B+ | Quadrature Encoder Signal |
| B- | Logical Complement of B+ |
| Z+ | Once Per Revolution |
| Z- | Logical Complement of Z+ |
| SHIELD | Case Ground |
| GROUND | Logical Ground |
| NC | No Connection |

*COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
MOTOR DRIVE SYSTEM TO TCS8 ENCODER SIGNALS

| DRAWING DATE | FILE NAME |
|---|---|
| JULY 3, 1991 | LANGLEY ENCODER |
| **DESIGN ENGINEER** | **DRAWING NUMBER** |
| TODD A. AMBUR | LFA-TAA-003 |




| SIGNAL | DESCRIPTION |
|---|---|
| HOME | Home Switch Signal |
| CW | End of Travel Limit Signal Clockwise |
| CCW | End of Travel Limit Signal Counter Clockwise |
| COMMON | Logical Ground |

*COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
MOTOR DRIVE SYSTEM TO LINEAR INDUSTRIES LIMIT SWITCHES

| | |
|---|---|
| DRAWING DATE<br>JULY 3, 1991 | FILE NAME<br>LANGLEY LIMIT SWITCH |
| DESIGN ENGINEER<br>TODD A. AMBUR | DRAWING NUMBER<br>LFA-TAA-004 |
| | |

**COMPLERE INC.**
**MOTOR DRIVE SYSTEM**

| SOCKET | SIGNAL |
|---|---|
| 1 | C1 +5V |
| 2 | C1 B+ |
| 3 | C1 SHIELD |
| 4 | C2 A+ |
| 5 | C2 GND |
| 6 | C3 +5V |
| 7 | C3 B+ |
| 8 | C3 SHIELD |
| 9 | C4 A+ |
| 10 | C4 GND |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | C1 A+ |
| 15 | C1 GND |
| 16 | C2 +5V |
| 17 | C2 B+ |
| 18 | C2 SHIELD |
| 19 | C3 A+ |
| 20 | C3 GND |
| 21 | C4 +5V |
| 22 | C4 B+ |
| 23 | C4 SHIELD |
| 24 | NC |
| 25 | NC |

DB-25 FEMALE

| PIN | SIGNAL |
|---|---|
| 1 | C1 +5V |
| 2 | C1 B+ |
| 3 | C1 SHIELD |
| 4 | C2 A+ |
| 5 | C2 GND |
| 6 | C3 +5V |
| 7 | C3 B+ |
| 8 | C3 SHIELD |
| 9 | C4 A+ |
| 10 | C4 GND |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | C1 A+ |
| 15 | C1 GND |
| 16 | C2 +5V |
| 17 | C2 B+ |
| 18 | C2 SHIELD |
| 19 | C3 A+ |
| 20 | C3 GND |
| 21 | C4 +5V |
| 22 | C4 B+ |
| 23 | C4 SHIELD |
| 24 | NC |
| 25 | NC |

DB-25 MALE

6 FT. DB-25 STRAIGHT THROUGH
EXTENSION CABLE

**TCS8**

| SOCKET | SIGNAL |
|---|---|
| 1 | C1 +5V |
| 2 | C1 B+ |
| 3 | C1 SHIELD |
| 4 | C2 A+ |
| 5 | C2 GND |
| 6 | C3 +5V |
| 7 | C3 B+ |
| 8 | C3 SHIELD |
| 9 | C4 A+ |
| 10 | C4 GND |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | C1 A+ |
| 15 | C1 GND |
| 16 | C2 +5V |
| 17 | C2 B+ |
| 18 | C2 SHIELD |
| 19 | C3 A+ |
| 20 | C3 GND |
| 21 | C4 +5V |
| 22 | C4 B+ |
| 23 | C4 SHIELD |
| 24 | NC |
| 25 | NC |

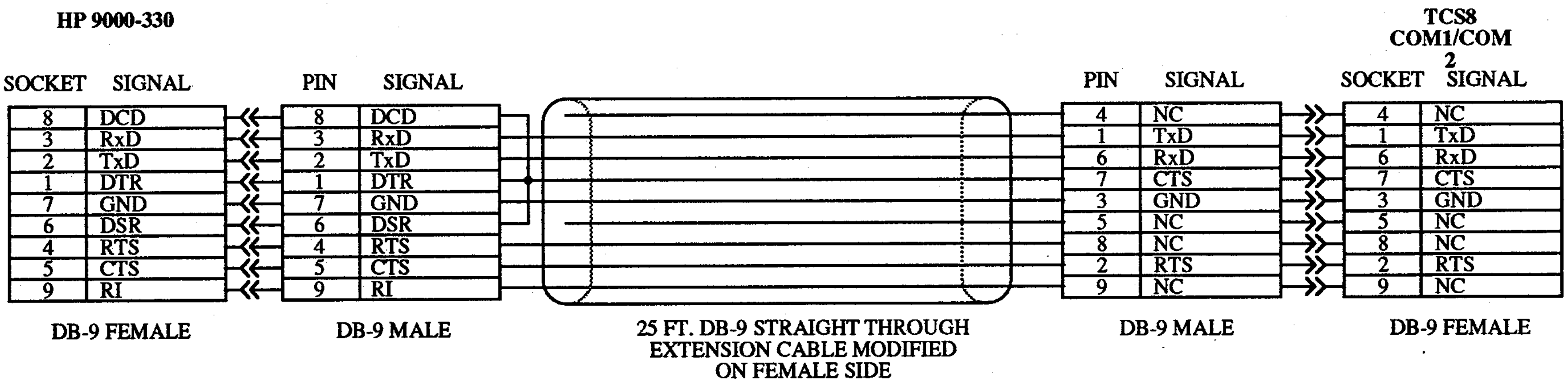
DB-25 FEMALE

| PIN | SIGNAL |
|---|---|
| 1 | NC |
| 2 | C1 B+ |
| 3 | C1 SHIELD |
| 4 | C2 A+ |
| 5 | C2 GND |
| 6 | NC |
| 7 | C3 B+ |
| 8 | C3 SHIELD |
| 9 | C4 A+ |
| 10 | C4 GND |
| 11 | NC |
| 12 | NC |
| 13 | NC |
| 14 | C1 A+ |
| 15 | C1 GND |
| 16 | NC |
| 17 | C2 B+ |
| 18 | C2 SHIELD |
| 19 | C3 A+ |
| 20 | C3 GND |
| 21 | NC |
| 22 | C4 B+ |
| 23 | C4 SHIELD |
| 24 | NC |
| 25 | NC |

DB-25 MALE

C1 - CHANNEL 1
C2 - CHANNEL 2
C3 - CHANNEL 3
C4 - CHANNEL 4

| | |
|---|---|
| *COMPLERE INC.* | |
| NASA LaRC LASER FLUORESCENCE ANEMOMETER<br>LFA TRAVERSE SYSTEM<br>MDS TO TCS8 ENCODER SIGNALS | |
| DRAWING DATE<br>JULY 3, 1991 | FILE NAME<br>ENCODER SIGNALS |
| DESIGN ENGINEER<br>TODD A. AMBUR | DRAWING NUMBER<br>LFA-TAA-005 |
| | |




| SIGNAL | DESCRIPTION |
|---|---|
| TxD | Transmit Data |
| RxD | Receive Data |
| RTS | Ready to Send |
| CTS | Clear to Send |
| DCD | Data Carrier Detect |
| DTR | Data Terminal Ready |
| DSR | Data Set Ready |
| RI | Ring Indicator |
| GND | Logical Ground |
| NC | No Connection |

| COMPLERE INC. | |
|---|---|
| NASA LaRC LASER FLUORESCENCE ANEMOMETER<br>LFA TRAVERSE SYSTEM<br>HP SERIES 9000 MODEL 330 TO TCS8 | |
| DRAWING DATE<br>JULY 3, 1991 | FILE NAME<br>HOST SERIAL |
| DESIGN ENGINEER<br>TODD A. AMBUR | DRAWING NUMBER<br>LFA-TAA-006 |

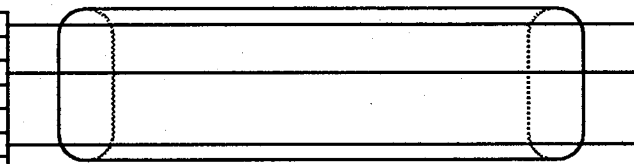
## TCS8 MDS1/MDS2

| SOCKET | SIGNAL |
|---|---|
| 1 | TxD |
| 2 | RTS |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | RxD |
| 7 | CTS |
| 8 | NC |
| 9 | NC |

DB-9 FEMALE

| PIN | SIGNAL |
|---|---|
| 1 | TxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | RxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 MALE



BELDEN CABLE 9925

| PIN | SIGNAL |
|---|---|
| 1 | TxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | RxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 MALE

## MDS HOST

| SOCKET | SIGNAL |
|---|---|
| 1 | RxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | TxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 FEMALE

| SIGNAL | DESCRIPTION |
|---|---|
| TxD | Transmit Data |
| RxD | Receive Data |
| RTS | Ready to Send |
| CTS | Clear to Send |
| GND | Logical Ground |
| NC | No Connection |

# *COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
TCS8 TO MDS SERIAL

| DRAWING DATE | FILE NAME |
|---|---|
| JULY 3, 1991 | TCS8 SERIAL |
| **DESIGN ENGINEER** | **DRAWING NUMBER** |
| TODD A. AMBUR | LFA-TAA-007 |

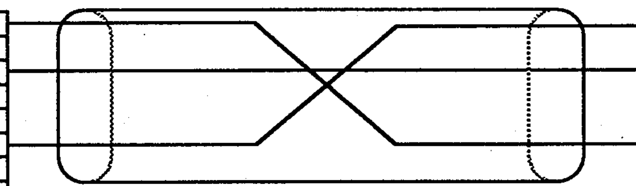
**MDS CHAIN**

| SOCKET | SIGNAL |
|---|---|
| 1 | RxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | TxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 FEMALE

| PIN | SIGNAL |
|---|---|
| 1 | RxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | TxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 MALE



BELDEN CABLE 9925

| PIN | SIGNAL |
|---|---|
| 1 | TxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | RxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 MALE

**MDS HOST**

| SOCKET | SIGNAL |
|---|---|
| 1 | RxD |
| 2 | NC |
| 3 | GND |
| 4 | NC |
| 5 | NC |
| 6 | TxD |
| 7 | NC |
| 8 | NC |
| 9 | NC |

DB-9 FEMALE

| SIGNAL | DESCRIPTION |
|---|---|
| TxD | Transmit Data |
| RxD | Receive Data |
| GND | Logical Ground |
| NC | No Connection |

*COMPLERE INC.*

NASA LaRC LASER FLUORESCENCE ANEMOMETER
LFA TRAVERSE SYSTEM
MDS TO MDS SERIAL

| DRAWING DATE | FILE NAME |
|---|---|
| JULY 3, 1991 | CHAIN SERIAL |
| **DESIGN ENGINEER** | **DRAWING NUMBER** |
| TODD A. AMBUR | LFA-TAA-008 |

## **Appendix B** Laser Velocimeter Data Acquisition System

The LVDAS acquires simultaneous digital data, analog data, and time information data. The data are sampled, multiplexed, buffered, and then transferred to the facility's host computer for further data reduction, analysis, and presentation.

Four 16 bit word parallel input ports are provided to accept the digital output of LV counter processors and/or other instrumentation.

New applications in laser velocimetry have brought about the need for a more advanced laser velocimeter data acquisition system. These new applications require high data rates that are not hindered by on-line time dependent data sorting and real time graphic data presentation. The new Laser Velocimeter Data Acquisition System (LVDAS) was designed specifically to meet these advanced requirements.

High data acquisition rates are achieved by providing a separate latched input for each laser velocimeter digital input and a separate converter for each laser fluorescence analog input. The system will allow for a data acquisition rates of approximately 100,000 samples per second simultaneously on each of the laser velocimeter and laser fluorescence inputs.

A 32 bit time of day (TOD) 10MHz counter is used to tag arrival times to acquired digital LDV data as they become available on each of four digital inputs. When a data valid sync pulse is sensed for a particular channel, the LVDAS latches the current TOD into a 32 bit time of arrival register (TOA). A separate TOA register is available for each digital input so that particle arrival times of measured velocity information U,V, W can be monitored for coincidence. The latched times of arrivals have a resolution of 100 ns and maximum time of over 7 minutes.

All of the acquired digital velocity data with corresponding time of arrival data can be processed and stored. However, if coincident data is required, then the arrival time of the various channels can be conditionally accepted if they all occur within a finite window of time. These coincident events can then be assigned interarrival times which represent elapsed time since the previous event.

The coincidence control logic allows for 3 channel coincidence. The coincidence time is adjustable from 0.1 μs to 1 s. In addition to the laser velocimeter inputs, three additional data words are generated internally. They are the inter arrival time, the coincidence time, and status words. The inter arrival and coincidence time is provided by a clock whose resolution and maximum time is 100 ns and 500 seconds respectively. The status word contains information about coincidence and the order in which the laser velocimeter data arrived.

During data acquisition, it is important that the user obtain some visual feedback about the data being acquired. This is necessary so that the user can make informed decisions about both the quality and quantity of data received. The user is either reassured about the quality of the data or can make alterations and improvements in technique while on line. To help achieve this, the instantaneous velocities are used to generate real time histograms from which probability density distributions are determined for all velocity components.

Additionally, the laser velocimeter data acquisition system has the capability of reducing the raw laser velocimeter data. Each laser velocimeter output contains the information required to

calculate the instantaneous velocities. From the instantaneous velocity determinations, the average velocities, turbulence levels, and the turbulence cross correlations are all be calculated.

The coincidence control logic will allow for up to 4 channel coincidence. The coincidence time can be adjustable to any resolution or duration within the capability of the time of arrival registers. When coincident criteria are met, the analog inputs can be sampled and converted to provide concurrent data with the digital data. A single time of arrival is latched for all four of the analog to digital inputs since they are all sampled and converted simultaneously. A final time of arrival is latched for external events. These might be derived from such sources as oscillating models or model surfaces, rotating helicopter blades, rotating engine fans, or flow sensors.

All digital Macrodyne data, optional digital data, analog to digital data, and time of arrival data can be sent by the LVDAS to other computers via two serial and two parallel input/output ports. One parallel port will be used for the HP series 9000 model 330 computer while the other can be used by the facility host computer. The serial ports can be used by PC type computers such as IBMs or MACs. Software has been developed for on-line data acquisition and display. A program listing is enclosed.

```
100 Main: !-----------------------------------------------------------------------------------------------!
110       !                                                                                               !
120       !          NASA LANGLEY RESEARCH CENTER                  Property of COMPLERE INC.              !
130       !           16 BY 24 INCH WATER TUNNEL                     Proprietary software                 !
140       !                                                        Copyright February 25, 1992            !
150       !          LASER FLUORESCENCE ANEMOMETER               Developed by: T. Kevin McDevitt          !
160       !                                                                                               !
170       !-----------------------------------------------------------------------------------------------!
180       !
190       ! PROGRAM DESCRIPTION:
200       !
210       !       This program provides the capability to acquire simultaneous Laser Doppler Velocimeter (LDV), Laser
220       !    Fluorescence Anemometer (LFA), and Analog Voltage Data at user selectable traverse controlled probe volume
230       !    positions within the water tunnel flow.
240       !       The LVDAS (Laser Velocimeter Data Acquisition System) is used to sample the LDV, LFA, and Analog Voltage
250       !    data simultaneously with a coincidence criterion being applied to LDV incoming data.  The LVDAS also generates
260       !    interarrival times and coincidence time.
270       !       The measured LDV data provides the necessary frequency information from which three components of flow
280       !    velocities can be determined.  These velocities are measured directly in "LASER" coordinates.  Coordinate
290       !    system transformations are applied to these measured velocities to obtain velocities in "TUNNEL" and "MODEL"
300       !    coordinates.
310       !       The TCS8 (Traverse Control System) is used to precisely move the LDV probe volume within the tunnel and
320       !    about the model.  The TCS8 provides three axes plus one auxiliary axis of traverse capability for both the
330       !    transmitting and receiving side optical packages.  The Tx and Rx side traverses can be moved independently to
340       !    achieve laser alignment or they can be moved together to maintain laser alignment.
350       !       The TCS8 will give the traverse positions in TCS8 coordinates where one inch of commanded movement will
360       !    yield one inch of movement on the traverse slides.  However, this will not yield one inch of movement of the
370       !    probe volume crossover point within the water filled tunnel because of the differences of refraction in air,
380       !    glass, and water.  Therefore, coordinate system transformations are applied to TCS8 positions to obtain
390       !    positions in "TUNNEL" and "MODEL" coordinates.
400       !       During data acquisition, real time histograms will be displayed of the LDV and analog data.  After the data
410       !    has been acquired, the averages, standard deviations, and shear stresses will be calculated and displayed in
420       !    profile plots where the data is plotted versus traverse position.  The reduced data is also sent to the
430       !    printer in tabular form.  The reduced data as well as the raw data are stored along with the tunnel conditions
440       !    on the hard disc for archival purposes and also to allow for further data reduction, data plotting, or data
450       !    transfer to other computers.
460       !
470       ! PROGRAM OPERATION:
480       !
490       !    The following power up sequences should be completed before this program is run:
500       !            1. Turn on the "Motor Drive System" boxes.
510       !            2. Turn on the "TCS8" traverse control system.
520       !            3. Turn on the "LVDAS" Laser Velocimeter Data Acquisition System.
530       !            4. Turn on the HP series 9000 model 330 computer.
540       !    This program will automatically be loaded and run when the computer is turned on.  If it is not loaded then
550       !    you can type in the following commands to load and then run it.
560       !            LOAD "LDVWT:,1400,0,0"
570       !            RUN
580       !    When the program is ready for user operation, it will display three things on the CRT. These are the main
590       !    menu, TCS8 traverse positions, and new sets of histogram & profile graphs.  If they do not appear on the CRT
600       !    then you should perform the following actions to reinitialize the systems.
610       !            1. Press shift reset on the HP series 9000 model 330 computers keyboard.
620       !            2. Press reset on the back of the TCS8.
630       !            3. Press reset on the front (or back) of the LVDAS.
640       !            4. LOAD "LDVWT:,1400,0,0"
650       !            5. RUN
660       !
670       ! PROGRAM VARIABLES:
680       !
690       ! Mass Storage Variables:
700       !
710       !    System$      Tells the program where to read/store system data related files.
720       !    Data$        Tells the program where to read/store raw & reduced data related files.
730       !    File$        File name for tunnel conditions data or raw & reduced data.
740       !
750       ! Menu Variables:
760       !
770       !    Menu$(*)     String array where each element describes its corresponding menu subroutine's function.
780       !    Menu         Used as an index to the string array Menu$(*).  Indicates which of the menus has been
790       !                 selected as the current menu.
800       !    Key          Used as an index to the string array Menu$(*).  Indicates which one of eight menu
810       !                 subroutines in the menu is to be executed.
820       !    Busy         Tells the Menu Status subprogram to display the current menu selection in inverse video.
830       !    Ready        Tells the Menu Status subprogram to display the current menu selection in normal text.
840       !
850       ! Traverse Position Variables:
860       !
870       !    Tcs1(*)      TCS8 transmitting side traverse positions (X1,Y1,Z1,A1) in TCS coordinates.
880       !    Tcs2(*)      TCS8  receiving   side traverse positions (X2,Y2,Z2,A2) in TCS coordinates.
890       !    Tun(*)       Traverse positions (X,Y,Z) in TUNNEL coordinates.
```

```
900      !     Mod(*)        Traverse positions (X,Y,Z) in MODEL  coordinates.
910      !
920      ! Auto Move Traverse Position Variables:
930      !
940      !     Pos(*)        Array of preprogrammed auto move positions.
950      !     Pname$(*)     Names for the variables in Pos(*).
960      !     Pimage$(*)    Image formats for the variables in Pos(*).
970      !     Punits$(*)    Units for the variables in Pos(*).
980      !     Npos          Number of preprogrammed auto move positions in Pos(*).
990      !     Paxis         Specifies which axis is to be traversed for the profile.  Also defines axis for plots.
1000     !
1010     ! Traverse Coordinate System Transformation Variables:
1020     !
1030     !     Index(*)      Array of indexes of refraction for air, glass, and water.
1040     !                       Index(1) : Index of refraction for Air.
1050     !                       Index(2) : Index of refraction for Glass.
1060     !                       Index(3) : Index of refraction for Water.
1070     !     Theta         Tx Side Off Axis Angle.
1080     !     Fs            Focal length for sending side onaxis and offaxis lenses.
1090     !     Fr            Focal length for receiving side offaxis lens.
1100     !     Bs            Beam spacings for sending side onaxis and offaxis beam pairs.
1110     !     Br            Beam spacing for receiving side offaxis.
1120     !     Ts            Angle of offaxis sending side beam pair.
1130     !     Tr            Angle of offaxis receiving side beam pair.
1140     !     Ta            Sending side offaxis auxiliary traverse angle.
1150     !     Tcs2tun1(*)   Sending   side coordinate system transformation matrix for converting Tcs1(*) to Tun(*).
1160     !     Tun2tcs1(*)   Sending   side coordinate system transformation matrix for converting Tun(*) to Tcs1(*).
1170     !     Tcs2tun2(*)   Receiving side coordinate system transformation matrix for converting Tcs2(*) to Tun(*).
1180     !     Tun2tcs2(*)   Receiving side coordinate system transformation matrix for converting Tun(*) to Tcs2(*).
1190     !     Tun2mod(*)    Coordinate system transformation matrix for converting Tun(*) to Mod(*).
1200     !     Mod2tun(*)    Coordinate system transformation matrix for converting Mod(*) to Tun(*).
1210     !
1220     ! Velocity Coordinate System Transformation Variables:
1230     !
1240     !     Index(*)      Array of indexes of refraction for air, glass, and water.
1250     !                       Index(1) : Index of refraction for Air.
1260     !                       Index(2) : Index of refraction for Glass.
1270     !                       Index(3) : Index of refraction for Water.
1280     !     Theta1(*)     Angles between LASER & TUNNEL UVW laser beams in Air (N=Index1).
1290     !     Theta3(*)     Angles between LASER & TUNNEL UVW laser beams in Water (N=Index3).
1300     !     Tun2ldv(*)    Coordinate system transformation matrix for converting from TUNNEL to LASER.
1310     !     Ldv2tun(*)    Coordinate system transformation matrix for converting from LASER to TUNNEL.
1320     !
1330     ! Traverse & Velocity Coordinate System Transformation Variables:
1340     !
1350     !     Alpha(*)      Angles of attack, yaw, and roll.
1360     !                       Alpha(1) : Angle of Attack.
1370     !                       Alpha(2) : Angle of Yaw.
1380     !                       Alpha(3) : Angle of Roll.
1390     !     Mod2tun(*)    Coordinate system transformation matrix for converting positions & velocities from MODEL to TUNNEL.
1400     !     Tun2mod(*)    Coordinate system transformation matrix for converting positions & velocities from TUNNEL to MODEL.
1410     !
1420     ! Tunnel Condition Variables:
1430     !
1440     !     Array(*)      Array of tunnel conditions, laser parameters, graph scales, etc.
1450     !     Name$(*)      Names for the variables in Array(*).
1460     !     Image$(*)     Image formats for the variables in Array(*).
1470     !     Units$(*)     Units for the variables in Array(*).
1480     !
1490     ! Misc. Tunnel Condition Variables:
1500     !
1510     !     Date          Date.
1520     !     Time          Time.
1530     !     Run           Run Number.
1540     !     File          File Number.
1550     !     Mach          Mach Number.
1560     !     Temp          Room Temperature (deg. F).
1570     !     Uedge         Freestream Velocity.
1580     !     Ujet_ue       Jet exit velocity normalized by Uedge.
1590     !
1600     ! LVDAS Variables:
1610     !
1620     !     Table(*)      Lookup table of frequencies.
1630     !     Atime         The maximum desired acquisition time (seconds).
1640     !     Ctime         The maximum desired coincidence time (seconds).
1650     !     At_exp        Exponent for interarrival times.
1660     !     Ct_exp        Exponent for coincidence times.
1670     !     Nreads        Number of desired samples.
1680     !     Nsam          Number of acquired samples.
1690     !     Coin(*)       Coincidence criteria.
```

```
1700    !    Cmask          Coincidence mask for U,V,W selection.
1710    !    Raw(*)         Array of raw data acquired from the LVDAS.
1720    !
1730    ! Instantaneous Velocity and Voltage Variables:
1740    !
1750    !    Ui(*)          Read from LVDAS as the instantaneous U frequency data, then converted into U velocities.
1760    !    Vi(*)          Read from LVDAS as the instantaneous V frequency data, then converted into V velocities.
1770    !    Wi(*)          Read from LVDAS as the instantaneous W frequency data, then converted into W velocities.
1780    !    Ai(*)          Read from LVDAS as the instantaneous A voltage data.
1790    !    Bi(*)          Read from LVDAS as the instantaneous B voltage data.
1800    !    Ii(*)          Read from LVDAS as the raw interarrival time data, then converted into interarrival times.
1810    !    Ci(*)          Read from LVDAS as the raw coincidence  time data, then converted into coincidence  times.
1820    !    Valid(*)       Validation words.  Initially all ones, then some set to zero during histogram clipping.
1830    !
1840    ! Histogram Clipping Variables:
1850    !
1860    !    Umin           The minimum acceptable U frequency (MHz).
1870    !    Umax           The maximum acceptable U frequency (MHz).
1880    !    Vmin           The minimum acceptable V frequency (MHz).
1890    !    Vmax           The maximum acceptable V frequency (MHz).
1900    !    Wmin           The minimum acceptable W frequency (MHz).
1910    !    Wmax           The maximum acceptable W frequency (MHz).
1920    !    Clip           Clip:  1 turn histogram clipping on; 0 turns it off.
1930    !
1940    ! Frequency to Velocity Conversion Variables:
1950    !
1960    !    Beam_spc(*)    Beam spacing at lens.
1970    !    Focl_len(*)    Focal length.
1980    !    Beam_sep(*)    Beam separation angle in degrees (full angle).
1990    !    Wave_len(*)    Wave length.
2000    !    Frng_spc(*)    Fringe Spacings.
2010    !    Brg_frq(*)     Bragg  Frequencies.
2020    !    Mix_frq(*)     Mixing Frequencies.
2030    !    Mea_sgn(*)     Measured Frequencies' Signs.
2040    !    Brg_sgn(*)     Bragg     Frequencies' Signs.
2050    !    Mix_sgn(*)     Mixing    Frequencies' Signs.
2060    !
2070    ! Summation Variables:
2080    !
2090    !    Sum(1,1)       Summation of all of the valid Ui.
2100    !    Sum(2,1)       Summation of all of the valid Vi.
2110    !    Sum(3,1)       Summation of all of the valid Wi.
2120    !    Sum(4,1)       Summation of all of the valid Ai.
2130    !    Sum(5,1)       Summation of all of the valid Bi.
2140    !    Sum(6,1)       Summation of all of the valid Ii.
2150    !    Sum(7,1)       Summation of all of the valid Ci.
2160    !    Sum(1,2)       Summation of all of the valid Ui*Ui.
2170    !    Sum(2,2)       Summation of all of the valid Vi*Vv.
2180    !    Sum(3,2)       Summation of all of the valid Wi*Ww.
2190    !    Sum(4,2)       Summation of all of the valid Ai*Ai.
2200    !    Sum(5,2)       Summation of all of the valid Bi*Bi.
2210    !    Sum(6,2)       Summation of all of the valid Ii*Ii.
2220    !    Sum(7,2)       Summation of all of the valid Ci*Ci.
2230    !    Sum(1,3)       Summation of all of the valid Ui*Vi.
2240    !    Sum(2,3)       Summation of all of the valid Vi*Wi.
2250    !    Sum(3,3)       Summation of all of the valid Wi*Ui.
2260    !    Sum(4,3)       Summation of all of the valid Ai*Bi.
2270    !    Sum(5,3)       Summation of all of the valid Ui*Ai.
2280    !    Sum(6,3)       Summation of all of the valid Vi*Ai.
2290    !    Sum(7,3)       Summation of all of the valid Wi*Ai.
2300    !    N(*)           Number of valid samples for the above summations.
2310    !
2320    ! Reduced Data Variables:
2330    !
2340    !    U              Average U frequency or velocity.
2350    !    V              Average V frequency or velocity.
2360    !    W              Average W frequency or velocity.
2370    !    A              Average A voltage.
2380    !    B              Average B voltage.
2390    !    I              Average interarrival time.
2400    !    C              Average coincidence time.
2410    !    U1             Standard deviation for U frequency or velocity.
2420    !    V1             Standard deviation for V frequency or velocity.
2430    !    W1             Standard deviation for W frequency or velocity.
2440    !    A1             Standard deviation for A voltage.
2450    !    B1             Standard deviation for B voltage.
2460    !    I1             Standard deviation for interarrival time.
2470    !    C1             Standard deviation for coincidence time.
2480    !    U1v1           Velocity:Velocity Shear Stress.
2490    !    V1w1           Velocity:Velocity Shear Stress.
```

```
2500     !     Wlul         Velocity:Velocity Shear Stress.
2510     !     Albl         Voltage :Voltage  Shear Stress.
2520     !     Ulal         Velocity:Voltage  Shear Stress.
2530     !     Vlal         Velocity:Voltage  Shear Stress.
2540     !     Wlal         Velocity:Voltage  Shear Stress.
2550     !
2560     ! Data Plotting Symbol Variables:
2570     !
2580     !     Symbols(*)   Array of Symbol arrays.  Each symbol arrays contains a distinct geometric symbol.
2590     !     Symbol(*)    Array of coordinates which when connected produce a distinct geometric symbol.
2600     !     Dot(*)       Array of coordinates which produce a dot.  The dot symbol is added to all symbols.
2610     !     Noc          The number of coordinates in a symbol.
2620     !     S            Used to index the Symbols array.
2630     !
2640     ! Histogram and Profile Graph Variables:
2650     !
2660     !     Wndw(*)      Array containing the plot's scales.
2670     !     Vwprt(*)     Array containing the plot's CRT position.
2680     !     Xdiv(*)      Array containing the number of X divisions for the plot's X axis.
2690     !     Ydiv(*)      Array containing the number of Y divisions for the plot's Y axis.
2700     !     Xlabel$(*)   String array containing labels for the X axis.
2710     !     Ylabel$(*)   String array containing labels for the Y axis.
2720     !     Title$(*)    String array containing labels for the Plots.
2730     !     Ximage$(*)   String array containing image formats for the X axis labeling.
2740     !     Yimage$(*)   String array containing image formats for the Y axis labeling.
2750     !     Legend$(*)   String array containing labels for each symbol in a profile plot.
2760     !     G            Used as an index to the above arrays.  Specifies one of nine plots.
2770     !     Gsave(*)     Used to save the entire graphics contents of the CRT.
2780     !
2790              OPTION BASE 1
2800              COM /Data/ INTEGER Raw(1000,10),Valid(1000),REAL Table(0:32766),Ui(1000),Vi(1000),Wi(1000),Ai(1000),
                             Bi(1000),Ii(1000),Ci(1000)
2810              COM /Array/ Name$(100,4)[10],Image$(100,4)[10],Units$(100,4)[10],REAL Array(100,4)
2820              COM /Pos/ Pname$(25,1)[10],Pimage$(25,1)[10],Punits$(25,1)[10],REAL Pos(25,1),Npos
2830              COM /Graph/ Wndw(9,4),Vwprt(9,4),Xdiv(9),Ydiv(9),Xlabel$(9)[80],Ylabel$(9)[80],Title$(9)[80],
                             Ximage$(9)[80],Yimage$(9)[80],Legend$(9,5)[80]
2840              COM Run,File,Paxis
2850              DIM Menu$(5,8)[80],System$[20],Data$[20],File$[50],L$[160]
2860              INTEGER Gsave(1280,1024),At_exp,Ct_exp,Cmask,Nsam,N(10,3)
2870              REAL Atime,Ctime,Sum(10,3),Symbols(5,0:20,3),Apos,Bpos
2880              DIM Tcs2tun1(4,4),Tun2tcs1(4,4),Tun2mod(3,3),Tun2ldv(3,3),Tun(4),Tcs1(4)
2890              DIM Tcs2tun2(4,4),Tun2tcs2(4,4),Mod2tun(3,3),Ldv2tun(3,3),Mod(4),Tcs2(4)
2900              DIM Beam_spc(3),Focl_len(3),Mea_sgn(3),Mix_frq(3),Mix_sgn(3),Frng_spc(3),Theta1(3,3)
2910              DIM Beam_sep(3),Wave_len(3),Brg_frq(3),Brg_sgn(3),Index(3),Coin(3),Theta3(3,3),Alpha(3)
2920              ! Perform trigonometric operations in degrees.
2930              DEG
2940              ! Clear the CRT and direct printed output to it.
2950              CLEAR SCREEN
2960              GCLEAR
2970              PRINTER IS CRT
2980              ! Perform any necessary setup and initialization routines.
2990              GOSUB Lvds_set_up        ! Initialize the HP to LVDAS interface.
3000              GOSUB File_set_up        ! Select mass storage device for system & data files.
3010              GOSUB Tcs8_set_up        ! Initialize the HP to TCS8  interface.
3020              GOSUB Menu_set_up        ! Initialize the user driven menus and display the main menu.
3030              GOSUB Grph_set_up        ! Initialize the CRT and plot the nine empty plots for profiles and histograms.
3040 Here:        ! The main program, while continually displaying the time of day, will wait hear for menu key selection.
3050              Date=TIMEDATE
3060              Time=Date
3070              DISP TIME$(Time),DATE$(Date)
3080              GOTO Here
3090 On_key:      ON KEY 1 GOSUB Key1    ! If the user function key #1 is ever pressed then execute the "Key1" subroutine.
3100              ON KEY 2 GOSUB Key2    ! If the user function key #2 is ever pressed then execute the "Key2" subroutine.
3110              ON KEY 3 GOSUB Key3    ! If the user function key #3 is ever pressed then execute the "Key3" subroutine.
3120              ON KEY 4 GOSUB Key4    ! If the user function key #4 is ever pressed then execute the "Key4" subroutine.
3130              ON KEY 5 GOSUB Key5    ! If the user function key #5 is ever pressed then execute the "Key5" subroutine.
3140              ON KEY 6 GOSUB Key6    ! If the user function key #6 is ever pressed then execute the "Key6" subroutine.
3150              ON KEY 7 GOSUB Key7    ! If the user function key #7 is ever pressed then execute the "Key7" subroutine.
3160              ON KEY 8 GOSUB Key8    ! If the user function key #8 is ever pressed then execute the "Key8" subroutine.
3170              RETURN
3180 Keys:        !  Subroutine Key1,Key2,Key3,Key4,Key5,Key6,Key7,Key8 descriptions:
3190              !        When one of the special user function keys is pressed, the main program will execute one the
3200              !    following eight subroutines.  Each of these subroutines performs essentially the same basic
3210              !    function in that it subsequently executes one of the menu subroutines.  The particular menu
3220              !    subroutine to be executed will depend on the current menu selected and the current key pressed.
3230              !        Before the selected menu subroutine is executed, the corresponding menu entry at the top of
3240              !    the CRT is redisplayed in inverse video.  This indicates that the menu selection has been
3250              !    acknowledged and that any resultant actions are still in progress.  When the highlighted menu
3260              !    subroutine has completed the current TCS8 traverse positions will be read and updated on the CRT
3270              !    display.  The corresponding menu entry displayed at the top of the CRT is redisplayed in normal
```

```
3280               !      text to indicate the completion of the menu subroutine.  The user can then select another special
3290               !      function key.
3300               !  Variables:
3310               !      Menu       Indicates which of the menus has been selected as the current menu.
3320               !      Key        Indicates which one of eight menu subroutines in the menu is to be executed.
3330               !      Menu$(*)   String array where each element describes its corresponding menu subroutine's function.
3340               !      Busy       Tells the Menu Status subroutine to display the current menu selection in inverse video.
3350               !      Ready      Tells the Menu Status subroutine to display the current menu selection in normal text.
3360 Key1:         Key=1
3370               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3380               ON Menu GOSUB M1k1,M2k1,M3k1,M4k1,M5k1,M6k1,M7k1
3390               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3400               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3410               RETURN
3420 Key2:         Key=2
3430               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3440               ON Menu GOSUB M1k2,M2k2,M3k2,M4k2,M5k2,M6k2,M7k2
3450               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3460               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3470               RETURN
3480 Key3:         Key=3
3490               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3500               ON Menu GOSUB M1k3,M2k3,M3k3,M4k3,M5k3,M6k3,M7k3
3510               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3520               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3530               RETURN
3540 Key4:         Key=4
3550               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3560               ON Menu GOSUB M1k4,M2k4,M3k4,M4k4,M5k4,M6k4,M7k4
3570               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3580               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3590               RETURN
3600 Key5:         Key=5
3610               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3620               ON Menu GOSUB M1k5,M2k5,M3k5,M4k5,M5k5,M6k5,M7k5
3630               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3640               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3650               RETURN
3660 Key6:         Key=6
3670               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3680               ON Menu GOSUB M1k6,M2k6,M3k6,M4k6,M5k6,M6k6,M7k6
3690               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3700               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3710               RETURN
3720 Key7:         Key=7
3730               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3740               ON Menu GOSUB M1k7,M2k7,M3k7,M4k7,M5k7,M6k7,M7k7
3750               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3760               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3770               RETURN
3780 Key8:         Key=8
3790               CALL Menu_status(Menu,Key,Busy,Menu$(*))
3800               ON Menu GOSUB M1k8,M2k8,M3k8,M4k8,M5k8,M6k8,M7k8
3810               CALL Menu_status(Menu,Key,Ready,Menu$(*))
3820               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
3830               RETURN
3840 Menu1:        !  Descriptions of the "Main Menu" subroutines M1K1,...,M1K8:
3850               !          The eight subroutines M1K1,...,M1K8 together implement the "Main Menu".  The following will be
3860               !      displayed at the top left of the CRT display when the "Main Menu" is selected:
3870               !
3880               !              M1K1: Laser Alignment
3890               !              M1K2: Pre Run
3900               !              M1K3: Post Run (Dump Graphics)
3910               !              M1K4: Set Auto Move Positions
3920               !              M1K5: Move traverse
3930               !              M1K6: Take data
3940               !              M1K7: Auto move and take
3950               !              M1K8: Display Histograms
3960               !
3970               !          M1K1 will change the current active menu from the "Main Menu" to the "Laser Alignment Menu".
3980               !      M1K2 will change the current active menu from the "Main Menu" to the "Pre Run Menu".  M1K3 will
3990               !      transfer the graphics contents of the CRT to the printer.  This provides a hard copy of the profile
4000               !      plots.  M1K4 has the user enter predefined traverse positions for a profile plot.  M1K5 moves the
4010               !      traverse to a user selectable position.  M1K6 acquires LVDAS data at the current TCS8 traverse
4020               !      position.  M1K7 acquires LVDAS data at each of the pre programed TCS8 traverse positions set up by
4030               !      M1K4.  M1K8 repeatedly displays five channels of real time histograms until the user presses any
4040               !      key on the keyboard.
4050               !
4060 M1k1:         ! Change the current active menu from the "Main Menu" to the "Laser Alignment Menu".
4070               Menu=2
```

```
4080                CALL Menu_disp(Menu,Menu$(*))
4090                RETURN
4100 M1k2:          ! Change the current active menu from the "Main Menu" to the "Pre Run Menu".
4110                Menu=3
4120                CALL Menu_disp(Menu,Menu$(*))
4130                RETURN
4140 M1k3:          ! Transfer the graphics contents of the CRT to the printer.  This provides a hard copy of the plots.
4150                KEY LABELS OFF                    ! Turn off the key labels so that they won't be printed.
4160                PRINTER IS CRT;WIDTH 132
4170                DISP ""                           ! Clear the CRT's display line so that they won't be printed.
4180                FOR L=1 TO 9                      ! Clear the CRT's menu lines so that it won't be printed.
4190                    PRINT TABXY(1,L);RPT$(" ",120)
4200                NEXT L
4210                PRINTER IS PRT
4220                PRINT USING "#,@"                 ! Move to the top of the next page on the printer.
4230                DUMP GRAPHICS                     ! Dump the entire CRT's contents to the printer.
4240                PRINT USING "#,@"                 ! Move to the top of the next page on the printer.
4250                PRINTER IS CRT
4260                CALL Menu_disp(Menu,Menu$(*))     ! Redisplay the menus.
4270                RETURN
4280 M1k4:          ! Have the user enter predefined traverse positions for a profile plot.
4290                CALL Enter_value("number of traverse positions",Npos,"K")
4300                REDIM Pos(Npos,1),Pname$(Npos,1),Pimage$(Npos,1),Punits$(Npos,1)
4310                MAT Pimage$= ("M4D.4D")
4320                MAT Punits$= ("in")
4330                FOR K=1 TO Npos
4340                    Pname$(K,1)="Pos#"&VAL$(K)
4350                NEXT K
4360                GSTORE Gsave(*)
4370                CALL Change("VALUES",Pos(*),Pname$(*),Pimage$(*),Punits$(*))
4380                GLOAD Gsave(*)
4390                CALL Menu_disp(Menu,Menu$(*))
4400                RETURN
4410 M1k5:          ! Moves the traverse to a user selectable position.
4420                GOSUB Read_calc_fill
4430                CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
4440                CALL Enter_value(CHR$(NUM("X")+Paxis-1),Mod(Paxis),"K")
4450 M1k5a:         ON KBD CALL Do_nothing
4460                DISP "Moving"
4470                Movement=Mod(Paxis)
4480                CALL Tcs8move(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Mod2tun(*),Tun2tcs1(*),Tun2tcs2(*),"Tx & Rx","MODEL",
                               "ABSOLUTE",Paxis,Movement)
4490                CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
4500                GOSUB Calc
4510                GOSUB Fill
4520                DISP ""
4530                OFF KBD
4540                RETURN
4550 M1k6:          ! Acquire LVDAS data at the current TCS8 traverse position.
4560              ! DISP "Press any key to TAKE DATA"
4570              ! CALL Rt_histo(@Lvdas,Symbols(*),1)
4580                Cmask=Coin(1)*1+Coin(2)*2+Coin(3)*4
4590                Nsam=MIN(Nreads,1000)
4600                Date=TIMEDATE
4610                Time=Date
4620                CALL Lvdas_take(@Lvdas,Atime,Ctime,At_exp,Ct_exp,Cmask,Nsam)
4630                IF Nsam>1 THEN
4640                    SELECT Paxis          ! Select the non-scanning axes for printing their position values at each point.
4650                    CASE 3
4660                        D$="X"
4670                        F$="Y"
4680                        Apos=Mod(1)
4690                        Bpos=Mod(2)
4700                    CASE 2
4710                        D$="X"
4720                        F$="Z"
4730                        Apos=Mod(1)
4740                        Bpos=Mod(3)
4750                    CASE 1
4760                        D$="Y"
4770                        F$="Z"
4780                        Apos=Mod(2)
4790                        Bpos=Mod(3)
4800                    END SELECT
4810                    OUTPUT PRT USING "K,K";CHR$(27)&"&k2S"&CHR$(27)&"&l9D",RPT$("=",140)
4820                    PRINTER IS PRT
4830                    Run$=VAL$(Run)
4840                    File$=VAL$(File)
4850                    PRINT " RUN "&Run$&" FILE "&File$
4860                    A$=DATE$(TIMEDATE)    ! Acquire the date and time for printing at each point.
```

```
4870                B$=TIME$(TIMEDATE)
4880                PRINT USING 4890;A$,B$,D$,Apos,F$,Bpos
4890                IMAGE 6X,11A,3X,8A,3X,A,"=",3D.4D,3X,A,"=",3D.4D
4900                PRINTER IS CRT
4910                CALL Data_reduce(At_exp,Ct_exp,Nsam)
4920                CALL Pt_histo(Symbols(*),Run,File,Mod(Paxis),Nsam)
4930                CALL Data_clip(Nsam,Umin,Umax,Vmin,Vmax,Wmin,Wmax)
4940                CALL Data_sum(Sum(*),N(*),Nsam)
4950                CALL Data_calc(N(*),Sum(*),U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,U1a1,V1a1,W1a1)
4960                B=U/U                   ! Replacement of B, B1, and A1b1 is being made by velocity ratios since
4970                B1=V/U                  !    the second analog channel B is not being used
4980                A1b1=W/U
4990                CALL Data_print(Paxis,Mod(Paxis),Nsam,"MHz",U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                          U1a1,V1a1,W1a1,Uedge)
5000                CALL Data_fconvert(Array(*))
5010                CALL Data_sum(Sum(*),N(*),Nsam)
5020                CALL Data_calc(N(*),Sum(*),U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,U1a1,V1a1,W1a1)
5030                B=U/Uedge
5040                B1=V/Uedge
5050                A1b1=W/Uedge
5060                CALL Data_print(Paxis,Mod(Paxis),Nsam,"LDV",U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                          U1a1,V1a1,W1a1,Uedge)
5070                CALL Data_trnsfrm(Ldv2tun(*),U,V,W,U1,V1,W1,U1v1,V1w1,W1u1,U1a1,V1a1,W1a1)
5080                B=U/Uedge
5090                B1=V/Uedge
5100                A1b1=W/Uedge
5110                CALL Data_print(Paxis,Mod(Paxis),Nsam,"TUN",U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                          U1a1,V1a1,W1a1,Uedge)
5120                CALL Data_trnsfrm(Tun2mod(*),U,V,W,U1,V1,W1,U1v1,V1w1,W1u1,U1a1,V1a1,W1a1)
5130                B=U/Uedge
5140                B1=V/Uedge
5150                A1b1=W/Uedge
5160                CALL Data_print(Paxis,Mod(Paxis),Nsam,"MOD",U,V,W,A,B,I0,C0,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                          U1a1,V1a1,W1a1,Uedge)
5170                CALL Data_plot(Array(*),Symbols(*),6,Mod(Paxis),U,V,W,1/Uedge,N(1,1),N(2,1),N(3,1))
5180                CALL Data_plot(Array(*),Symbols(*),7,Mod(Paxis),U1,V1,W1,1/Uedge,N(1,2),N(2,2),N(3,2))
5190                CALL Data_plot(Array(*),Symbols(*),8,Mod(Paxis),U1v1,V1w1,W1u1,1/Uedge^2,N(1,3),N(2,3),N(3,3))
5200                CALL Data_plot(Array(*),Symbols(*),9,Mod(Paxis),A,A,A1,1,N(4,1),N(4,1),N(4,2))
5210                OUTPUT PRT USING "K,K";CHR$(27)&"&k2S"&CHR$(27)&"&l9D",RPT$("=",140)
5220                GOSUB Store_file
5230                File=File+1
5240            END IF
5250            RETURN
5260 M1k7:      ! Acquire LVDAS data at each of the pre programed TCS8 traverse positions set up by M1K4.
5270            Quit=0
5280            ON KBD GOSUB Quit
5290            FOR J=1 TO Npos
5300                CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
5310                Mod(Paxis)=Pos(J,1)
5320                GOSUB M1k5a
5330                GOSUB M1k6
5340                IF Quit THEN 5360
5350            NEXT J
5360            OFF KBD
5370            GOSUB On_key
5380            CALL Menu_disp(Menu,Menu$(*))
5390            RETURN
5400 M1k8:      ! Repeatedly displays five channels of real time histograms until the user presses any key on the keyboard.
5410            DISP "Press any key to return to main menu"
5420            CALL Rt_histo(@Lvdas,Symbols(*),1)
5430            RETURN
5440 Menu2:     !  Descriptions of the "Laser Alignment Menu" subroutines M2K1,...,M2K8:
5450            !        The eight subroutines M2K1,...,M2K8 together implement the "Laser Alignment Menu".  The
5460            !     following will be displayed at the top left of the CRT display when the "Laser Alignment Menu" is
5470            !     selected:
5480            !
5490            !             M2K1: Return to main menu
5500            !             M2K2: Sides       : Tx & Rx
5510            !             M2K3: Coordinates: MODEL
5520            !             M2K4: Mode        : ABSOLUTE
5530            !             M2K5: Move X
5540            !             M2K6: Move Y
5550            !             M2K7: Move Z
5560            !             M2K8: Move A
5570            !
5580            !        M2K1 will change the current active menu from the "Laser Alignment Menu" to the "Main Menu".
5590            !     M2K2 selects whether the transmitting, receiving, or both sides of the traverse are to be moved.
5600            !     M2K3 selects the TCS, TUNNEL, or MODEL coordinate systems for traverse movements.  M2K4
5610            !     specifies movements to be relative to the currents position or to absolute positions.  M2K5 has the
5620            !     user enter a movement for the X axis and then the movement is performed.  M2K6 has the user enter
```

```
5630               !       a movement for the Y axis and then the movement is performed.  M2K7 has the user enter a movement
5640               !       for the Z axis and then the movement is performed.  M2K8 has the user enter a movement for the A
5650               !       axis and then the movement is performed.
5660               !
5670 M2k1:         ! Change the current active menu from the "Laser Alignment Menu" to the "Main Menu".
5680               Menu=1
5690               CALL Menu_disp(Menu,Menu$(*))
5700               RETURN
5710 M2k2:         ! Select whether the transmitting, receiving, or both sides of the traverse are to be moved.
5720               SELECT TRIM$(Menu$(Menu,Key)[20])
5730               CASE "Tx & Rx"
5740                 Menu$(Menu,Key)[20]="Tx"
5750               CASE "Tx"
5760                 Menu$(Menu,Key)[20]="Rx"
5770               CASE "Rx"
5780                 Menu$(Menu,Key)[20]="Tx & Rx"
5790               END SELECT
5800               CALL Menu_disp(Menu,Menu$(*))
5810               RETURN
5820 M2k3:         ! Selects the TCS, TUNNEL, or MODEL coordinate systems for traverse movements.
5830               SELECT TRIM$(Menu$(Menu,Key)[20])
5840               CASE "MODEL"
5850                 Menu$(Menu,Key)[20]="TUNNEL"
5860               CASE "TUNNEL"
5870                 Menu$(Menu,Key)[20]="TCS"
5880               CASE "TCS"
5890                 Menu$(Menu,Key)[20]="MODEL"
5900               END SELECT
5910               CALL Menu_disp(Menu,Menu$(*))
5920               RETURN
5930 M2k4:         ! Specifies movements to be relative to the currents position or to absolute positions.
5940               SELECT TRIM$(Menu$(Menu,Key)[20])
5950               CASE "ABSOLUTE"
5960                 Menu$(Menu,Key)[20]="RELATIVE"
5970               CASE "RELATIVE"
5980                 Menu$(Menu,Key)[20]="ABSOLUTE"
5990               END SELECT
6000               CALL Menu_disp(Menu,Menu$(*))
6010               RETURN
6020 M2k5:         !           The subroutines M2K5 thru M2K8 all execute the same code.  The code will have the user enter a
6030 M2k6:         !       movement for the X,Y,Z, or A depending on what the value of "Key" is.  The user specified movement
6040 M2k7:         !       for the selected axis will then be performed.
6050 M2k8:         !
6060               Side$=TRIM$(Menu$(Menu,2)[20])
6070               Coor$=TRIM$(Menu$(Menu,3)[20])
6080               Mode$=TRIM$(Menu$(Menu,4)[20])
6090               CALL Enter_value(Mode$&" Movement",Movement,"4D.5D")
6100               ON KBD CALL Do_nothing
6110               DISP "Moving"
6120               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
6130               CALL Tcs8move(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Mod2tun(*),Tun2tcs1(*),Tun2tcs2(*),Side$,Coor$,Mode$,
                               Key-4,Movement)
6140               CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
6150               DISP ""
6160               OFF KBD
6170               RETURN
6180 Menu3:        !  Descriptions of the "Pre Run Menu" subroutines M3K1,...,M3K8:
6190               !          The eight subroutines M3K1,...,M3K8 together implement the "Pre Run Menu".  The following will
6200               !      be displayed at the top left of the CRT display when the "Pre Run Menu" is selected:
6210               !
6220               !              M3K1: Return to MAIN menu
6230               !              M3K2: Enter Run & File Numbers
6240               !              M3K3: Enter Number of Samples
6250               !              M3K4: Select Traverse Axis for Profile
6260               !              M3K5: Print Coordinate Transformation Matrices
6270               !              M3K6: Setup Graphics
6280               !              M3K7: Tunnel Conditions
6290               !              M3K8: Traverse
6300               !
6310               !          M3K1 will change the current active menu from the "Pre Run Menu" to the "Main Menu".  M3K2 has
6320               !      the user enter a the Run and File numbers.  A new run number should be assigned to each profile
6330               !      while a new file number is assigned to each set of data.  M3K3 has the user enter the desired
6340               !      number of samples.  M3K4 has the user select which axis to traverse in for the profiles.  M3K5
6350               !      prints the coordinate system transformation matrices for both traverse positions and velocities.
6360               !      M3K6 creates a new set of empty plots for new profiles.  M3K7 will change the current active menu
6370               !      from the "Pre Run Menu" to the "Tunnel Conditions Menu".  M3K8 will change the current active menu
6380               !      from the "Pre Run Menu" to the "Traverse Menu".
6390               !
6400 M3k1:         ! Change the current active menu from the "Pre Run Menu" to the "Main Menu".
6410               Menu=1
```

```
6420           CALL Menu_disp(Menu,Menu$(*))
6430           RETURN
6440 M3k2:     ! Have the user enter a the Run and File numbers.
6450           CALL Enter_value("Run",Run,"3D.2D")
6460           CALL Enter_value("File",File,"3D")
6470           RETURN
6480 M3k3:     ! Have the user enter the desired number of samples.
6490           CALL Enter_value("Number of Samples ",Nreads,"K")
6500           RETURN
6510 M3k4:     ! Have the user select which axis to traverse in for the profiles.
6520           CALL Enter_string("Traverse Axis for Profiles ",Paxis$,"K")
6530           SELECT Paxis$
6540           CASE "X"
6550               Paxis=1
6560           CASE "Y"
6570               Paxis=2
6580           CASE "Z"
6590               Paxis=3
6600           CASE "A"
6610               Paxis=4
6620           CASE ELSE
6630               GOTO M3k4
6640           END SELECT
6650           GOSUB Fill
6660           RETURN
6670 M3k5:     ! Prints the coordinate system transformation matrices for both traverse positions and velocities.
6680           GOSUB Read_calc_fill
6690           OUTPUT PRT USING "#,2/"
6700           OUTPUT PRT USING "20X,K,/";"TRAVERSE COORDINATE TRANSFORMATION MATRICES"
6710           OUTPUT PRT USING "20X,K,/,4(13X,4(8D.5D),/)";"Transmitting side TCS8 to TUNNEL",Tcs2tun1(*)
6720           OUTPUT PRT USING "20X,K,/,4(13X,4(8D.5D),/)";"Receiving side TCS8 to TUNNEL",Tcs2tun2(*)
6730           OUTPUT PRT USING "20X,K,/,4(13X,4(8D.5D),/)";"Transmitting side TUNNEL to TCS8",Tun2tcs1(*)
6740           OUTPUT PRT USING "20X,K,/,4(13X,4(8D.5D),/)";"Receiving side TUNNEL to TCS8",Tun2tcs2(*)
6750           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"TUNNEL to MODEL",Tun2mod(*)
6760           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"MODEL to TUNNEL",Mod2tun(*)
6770           OUTPUT PRT USING "20X,K,/";"VELOCITY COORDINATE TRANSFORMATION MATRICES"
6780           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"LASER to TUNNEL",Ldv2tun(*)
6790           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"TUNNEL to LASER",Tun2ldv(*)
6800           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"TUNNEL to MODEL",Tun2mod(*)
6810           OUTPUT PRT USING "20X,K,/,3(13X,3(8D.5D),/)";"MODEL to TUNNEL",Mod2tun(*)
6820           OUTPUT PRT USING "#,@"
6830           RETURN
6840 M3k6:     ! Display a new set of plots for new profiles.
6850           CALL Setup_graph(Array(*),Image$(*),Paxis,Symbols(*))
6860           RETURN
6870 M3k7:     ! Change the current active menu from the "Pre Run Menu" to the "Tunnel Conditions Menu".
6880           Menu=4
6890           CALL Menu_disp(Menu,Menu$(*))
6900           RETURN
6910 M3k8:     ! Change the current active menu from the "Pre Run Menu" to the "Traverse Menu".
6920           Menu=5
6930           CALL Menu_disp(Menu,Menu$(*))
6940           RETURN
6950 Menu4:    !  Descriptions of the "Tunnel Conditions Menu" subroutines M4K1,...,M4K8:
6960           !        The eight subroutines M4K1,...,M4K8 together implement the "Tunnel Conditions Menu".  The
6970           !    following will be displayed at the top left of the CRT display when the "Tunnel Conditions Menu" is
6980           !    selected:
6990           !
7000           !            M4K1: Return to PRE RUN menu
7010           !            M4K2: Load  Tunnel Conditions
7020           !            M4K3: Save  Tunnel Conditions
7030           !            M4K4: Print Tunnel Conditions
7040           !            M4K5: Enter Tunnel Condition Data
7050           !            M4K6: Enter Tunnel Condition Names
7060           !            M4K7: Enter Tunnel Condition Units
7070           !            M4K8: Enter Tunnel Condition Images
7080           !
7090           !        M4K1 will change the current active menu from the "Tunnel Conditions Menu" to the "Pre Run
7100           !    Menu".  M4K2 loads the old tunnel conditions from a file on the disk.  M4K3 saves the current
7110           !    tunnel conditions to a file on the disk.  M4K2 & M4K3 load and save default tunnel conditions from
7120           !    the file "ARRAY" on the hard disk.  The default values are not related to any particular run number.
7130           !    M4K4 sends the current tunnel conditions to the printer.  M4K5 has the user enter values for the
7140           !    tunnel condition variables.  M4K6 has the user enter names for the tunnel condition variables.
7150           !    M4K7 has the user enter units for the tunnel condition variables.  M4K8 has the user enter image
7160           !    formats for the tunnel condition variables.
7170           !
7180 M4k1:     ! Change the current active menu from the "Tunnel Conditions Menu" to the "Pre Run Menu".
7190           Menu=3
7200           CALL Menu_disp(Menu,Menu$(*))
7210           RETURN
```

```
7220 M4k2:      ! Load the old tunnel conditions from a file on the disk.  This loads the default values.
7230            GOSUB Read_array
7240            GOSUB Read_calc_fill
7250            RETURN
7260 M4k3:      ! Save the current tunnel conditions to a file on the disk.  This updates the default values on the disk.
7270            GOSUB Read_calc_fill
7280            GOSUB Save_array
7290            RETURN
7300 M4k4:      ! Print the current tunnel conditions.
7310            GOSUB Read_calc_fill
7320            GOSUB Print_header
7330            RETURN
7340 M4k5:      ! Have the user enter values for the tunnel condition variables.
7350            GSTORE Gsave(*)
7360            GOSUB Read_calc_fill
7370            CALL Change("VALUES",Array(*),Name$(*),Image$(*),Units$(*))
7380            GOSUB Read_calc_fill
7390            GLOAD Gsave(*)
7400            RETURN
7410 M4k6:      ! Have the user enter names for the tunnel condition variables.
7420            GSTORE Gsave(*)
7430            GOSUB Read_calc_fill
7440            CALL Change("NAMES",Array(*),Name$(*),Image$(*),Units$(*))
7450            GOSUB Read_calc_fill
7460            GLOAD Gsave(*)
7470            RETURN
7480 M4k7:      ! Have the user enter units for the tunnel condition variables.
7490            GSTORE Gsave(*)
7500            GOSUB Read_calc_fill
7510            CALL Change("UNITS",Array(*),Name$(*),Image$(*),Units$(*))
7520            GOSUB Read_calc_fill
7530            GLOAD Gsave(*)
7540            RETURN
7550 M4k8:      ! Have the user enter image formats for the tunnel condition variables.
7560            GSTORE Gsave(*)
7570            GOSUB Read_calc_fill
7580            CALL Change("IMAGES",Array(*),Name$(*),Image$(*),Units$(*))
7590            GOSUB Read_calc_fill
7600            GLOAD Gsave(*)
7610            RETURN
7620 Menu5:     !  Descriptions of the "Traverse Menu" subroutines M5K1,...,M5K8:
7630            !         The eight subroutines M5K1,...,M5K8 together implement the "Traverse Menu".  The following will
7640            !     be displayed at the top left of the CRT display when the "Traverse Menu" is selected:
7650            !
7660            !             M5K1: Return to PRE RUN menu
7670            !             M5K2: View & Set TCS8 Positions
7680            !             M5K3: View & Set TCS8 Units
7690            !             M5K4: View & Set TCS8 Revolution
7700            !             M5K5: View & Set TCS8 Velocity
7710            !             M5K6: View & Set TCS8 Acceleration
7720            !             M5K7:
7730            !             M5K8:
7740            !
7750            !         M5K1 will change the current active menu from the "Traverse Menu" to the "Pre Run Menu".  M5K2
7760            !     reads from the TCS8 the current positions and lets the user change them.  The new positions are
7770            !     then sent to the TCS8.  M5K3 reads from the TCS8 the current counts per unit length (inches) and
7780            !     lets the user change them.  The new counts per unit length are then sent to the TCS8.  M5K4 reads
7790            !     from the TCS8 the current counts per revolution and lets the user change them.  The new counts per
7800            !     revolution are then sent to the TCS8.  M5K5 reads from the TCS8 the current velocities and lets the
7810            !     user change them.  The new velocities are then sent to the TCS8.  M5K6 reads from the TCS8 the
7820            !     current accelerations and lets the user change them.  The new accelerations are then sent to the
7830            !     TCS8.  M5K7 does nothing.  M5K8 does nothing.
7840            !
7850 M5k1:      ! Change the current active menu from the "Traverse Menu" to the "Pre Run Menu".
7860            Menu=3
7870            CALL Menu_disp(Menu,Menu$(*))
7880            RETURN
7890 M5k2:      ! Read current TCS8 positions, have the user update them, & then send the new values to the TCS8.
7900            CALL Tcs8set("P",@Tcs8)          ! View and set TCS8 Positions.
7910            GRAPHICS ON
7920            CALL Menu_disp(Menu,Menu$(*))
7930            RETURN
7940 M5k3:      ! Read current TCS8 counts per inch, have the user update them, & then send the new values to the TCS8.
7950            CALL Tcs8set("U",@Tcs8)          ! View and set TCS8 counts per Unit length.
7960            GRAPHICS ON
7970            CALL Menu_disp(Menu,Menu$(*))
7980            RETURN
7990 M5k4:      ! Read current TCS8 counts per revolution, have the user update them, & then send new values to the TCS8.
8000            CALL Tcs8set("R",@Tcs8)          ! View and set TCS8 counts per Revolution.
8010            GRAPHICS ON
```

```
8020                CALL Menu_disp(Menu,Menu$(*))
8030                RETURN
8040 M5k5:          ! Read current TCS8 velocities, have the user update them, & then send the new values to the TCS8.
8050                CALL Tcs8set("V",@Tcs8)          ! View and set TCS8 Velocities.
8060                GRAPHICS ON
8070                CALL Menu_disp(Menu,Menu$(*))
8080                RETURN
8090 M5k6:          ! Read current TCS8 accelerations, have the user update them, & then send the new values to the TCS8.
8100                CALL Tcs8set("A",@Tcs8)          ! View and set TCS8 Accelerations.
8110                GRAPHICS ON
8120                CALL Menu_disp(Menu,Menu$(*))
8130                RETURN
8140 M5k7:          RETURN   ! This subroutine does nothing.
8150 M5k8:          RETURN   ! This subroutine does nothing.
8160 Quit:          Quit=1   ! Quit will be set during a multiple position scan (see M1K7) if any key is pressed on the
8170                RETURN   ! keyboard during the scan.  This indicates that the scan should be terminated.
8180 Lvds_set_up:   ! This subroutine initializes the HP to LVDAS high speed parallel interface.  A communications path named
8190                ! "@Lvdas" is opened.  Also, this subroutine creates the raw data to frequency conversion look up table.
8200                CALL Lvdas_init(@Lvdas)
8210                CALL Table(Table(*))
8220                RETURN
8230 File_set_up:   ! This subroutine reads the initialization files from the disk.  System$ tells the program where to read
8240                ! system related files while Data$ tells the program where to read and store raw and reduced data.
8250                System$=":,1400,0,0"
8260                Data$=":,1400,0,1"
8270                LOAD KEY "KEYS"&System$
8280                GOSUB Read_array
8290                GOSUB Read_calc_fill
8300                GOSUB Save_array
8310                CLEAR SCREEN
8320                RETURN
8330 Tcs8_set_up:   ! This subroutine initialized the HP to TCS8 serial interface.  The communications path "@Tcs8" is opened.
8340                CALL Tcs8init(@Tcs8)
8350                CALL Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
8360                GOSUB Calc
8370                GOSUB Fill
8380                RETURN
8390 Grph_set_up:   ! This subroutine defines the graphics symbols for plotting data points, clears and initializes the CRT,
8400                ! and displays a new empty set of graphs for histogram and profile plotting.
8410                CALL Read_symbols(Symbols(*))
8420                CALL Crt_init
8430                CALL Setup_graph(Array(*),Image$(*),Paxis,Symbols(*))
8440                RETURN
8450 Menu_set_up:   ! This subroutine defines the menu descriptors for all of the menus.  The current menu is set to the "Main
8460                ! Menu" and its menu is displayed at the top of the screen.
8470                CALL Menu_read(Menu$(*))
8480                CALL Menu_disp(Menu,Menu$(*))
8490                GOSUB On_key
8500                Busy=0
8510                Ready=1
8520                RETURN
8530 Print_header:  ! This subroutine prints a header on the printers paper.  The "header" is a formatted list of all of the
8540                ! tunnel conditions, laser parameters, and graph scales.
8550                PRINTER IS PRT;WIDTH 144
8560                PRINT USING "#,5(K)";CHR$(27)&"&k2S"&CHR$(27)&"&l9D"
8570                CALL Array_print(Array(*),Name$(*),Image$(*),Units$(*))
8580                PRINT USING "#,@,5(K)";CHR$(27)&"E"
8590                PRINTER IS CRT
8600                RETURN
8610 Read_calc_fill: ! This subroutine extracts (reads) the tunnel conditions from the Array(*).  These values can be used to
8620                ! calculate other tunnel conditions.  The original tunnel conditions along with any calculated tunnel
8630                ! conditions are then put back (filled) into the Array(*).
8640                GOSUB Read
8650                GOSUB Calc
8660                GOSUB Fill
8670                RETURN
8680 Store_header:  ! This subroutine stores the header Array(*) and other arrays onto the disk.  There will be one header
8690                ! file for each run number.  For example, if the run number equal 1, then the data will be stored in a
8700                ! file named "R1".  This file will include an extensive list of tunnel conditions, laser parameters, graph
8710                ! scales, traverse positions, coordinate system transformation matrices, etc.
8720                DISP "Storing Header"
8730                ! Set File$ equal to the file name for the header file.  Each run number will have a different file name.
8740                File$="R"&VAL$(Run)&Data$
8750                ! Check if the file already exists.  If it does then, ask the user if he wants to overwrite the old file.
8760                ON ERROR GOTO 8960
8770                ASSIGN @Data TO File$
8780                OFF ERROR
8790                FOR K=1 TO 10
8800                    WAIT .2
8810                    BEEP
```

```
8820            NEXT K
8830            INPUT "Over Write old file? (Y or N) ",L$
8840            SELECT L$[1,1]
8850            CASE "Y","y"          ! If the user wants to overwrite the old file, then purge the old file.
8860                ASSIGN @Data TO *
8870                PURGE File$
8880                GOTO 8960
8890            CASE "N","n"          ! If the user doesn't want to overwrite the old file, then have a new run# entered.
8900                CALL Enter_value("Run",Run,"3D.2D")
8910                CALL Enter_value("File",File,"3D")
8920                GOTO Store_header
8930            CASE ELSE
8940                GOTO Store_header
8950            END SELECT
8960            OFF ERROR
8970            Fsize=INT((3200+4000*3+128*4+72*4)/256*1.05+1)           ! Calculate the headers file size.
8980            CREATE BDAT File$,Fsize                                  ! Create the header's file.
8990            ASSIGN @Data TO File$                                    ! Open the header's file.
9000            OUTPUT @Data;Array(*),Name$(*),Image$(*),Units$(*)
9010            OUTPUT @Data;Tun2tcs1(*),Tun2tcs2(*),Mod2tun(*),Tun2ldv(*)
9020            OUTPUT @Data;Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*),Ldv2tun(*)
9030            ASSIGN @Data TO *                                        ! Close the header's file.
9040            RETURN
9050 Store_file:    ! This subroutine stores the header Array(*), the raw data, and the reduced data onto the disk.  There
9060            ! will be one data file for each data set.  For example, if the run and file numbers equal 7 and 5
9070            ! respectively, then the data will be stored in a file named "R7F5".
9080            GOSUB Calc    ! Use the tunnel conditions to calculate and/or update other tunnel conditions.
9090            GOSUB Fill    ! Fill Array(*) with the original tunnel conditions along with the updated tunnel conditions.
9100            IF File=1 THEN GOSUB Store_header
9110            DISP "Storing Data"
9120            File$="R"&VAL$(Run)&"F"&VAL$(File)&Data$
9130            ! Check if the file already exists.  If it does, then ask the user if he wants to overwrite the old file.
9140            ON ERROR GOTO 9340
9150            ASSIGN @Data TO File$
9160            OFF ERROR
9170            FOR K=1 TO 10
9180                WAIT .2
9190                BEEP
9200            NEXT K
9210            INPUT "Over Write old file? (Y or N) ",L$
9220            SELECT L$[1,1]
9230            CASE "Y","y"          ! If the user wants to overwrite the old file, then purge the old file.
9240                ASSIGN @Data TO *
9250                PURGE File$
9260                GOTO 9340
9270            CASE "N","n"          ! If the user doesn't want to overwrite the old file, then have a new run# entered.
9280                CALL Enter_value("Run",Run,"3D.2D")
9290                CALL Enter_value("File",File,"3D")
9300                GOTO Store_file
9310            CASE ELSE
9320                GOTO Store_file
9330            END SELECT
9340            OFF ERROR
9350            Fsize=INT((3200+Nsam*10*2+60+240)/256*1.05+1)            ! Calculate the data's file size.
9360            CREATE BDAT File$,Fsize                                  ! Create the data's file.
9370            ASSIGN @Data TO File$                                    ! Open the data's file.
9380            OUTPUT @Data;Array(*),Raw(*),N(*),Sum(*)
9390            ASSIGN @Data TO *                                        ! Close the data's file.
9400            RETURN
9410 Read_array:    ! This subroutine reads the header Array(*) off of the disk from a file named "ARRAY".  The file will the
9420            ! have default values for the tunnel conditions, laser parameters, graph scales, etc.  This file is not
9430            ! meant to be attached to any run number or profile scan.  It is used to provide default values for the
9440            ! program so that the user will not have to enter a rather lengthy list of tunnel conditions.
9450            ON ERROR GOTO 9550
9460            ! If the file already exists, then read the Array(*) from the disk.
9470            ASSIGN @File TO "ARRAY"&System$
9480            ENTER @File;Array(*),Name$(*),Image$(*),Units$(*)
9490            ENTER @File;Tun2tcs1(*),Tun2tcs2(*),Mod2tun(*),Tun2ldv(*)
9500            ENTER @File;Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*),Ldv2tun(*)
9510            ASSIGN @File TO *
9520            OFF ERROR
9530            RETURN
9540            ! If the file doesn't exist then create the file, read in default data, and store the Array(*) on disk.
9550            OFF ERROR
9560            ASSIGN @File TO *
9570            ON ERROR GOTO 9590
9580            PURGE "ARRAY"&System$
9590            OFF ERROR
9600            CALL Array_init(Name$(*),Array(*),Image$(*),Units$(*))
9610            CREATE BDAT "ARRAY"&System$,50
```

```
9620                GOSUB Save_array
9630                RETURN
9640 Save_array:    ! This subroutine saves the header Array(*) onto the disk in a file named "ARRAY".  The file will then have
9650                ! default values for the tunnel conditions, laser parameters, graph scales, etc.  This file is not meant to
9660                ! be attached to any run number or profile scan.  It is used to provide default values for the program so
9670                ! that the user will not have to enter a rather lengthy list of tunnel conditions.
9680                ASSIGN @File TO "ARRAY"&System$
9690                OUTPUT @File;Array(*),Name$(*),Image$(*),Units$(*)
9700                OUTPUT @File;Tun2tcs1(*),Tun2tcs2(*),Mod2tun(*),Tun2ldv(*)
9710                OUTPUT @File;Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*),Ldv2tun(*)
9720                ASSIGN @File TO *
9730                RETURN
9740 Fill:          ! This subroutine fills the Array(*) with the current tunnel conditions, laser parameters, and histogram
9750                ! & profile scales.
9760                Array(1,1)=Date                  ! Date.
9770                Array(1,2)=Mach                  ! Mach Number.
9780                Array(1,4)=Alpha(1)              ! Angle of Attack.
9790                Array(2,1)=Time                  ! Time.
9800                Array(2,2)=Temp                  ! Room Temperature (deg. F).
9810                Array(2,4)=Alpha(2)              ! Angle of Yaw.
9820                Array(3,1)=Run                   ! Run Number.
9830                Array(3,2)=Uedge                 ! Freestream Velocity.
9840                Array(3,4)=Alpha(3)              ! Angle of Roll.
9850                Array(4,1)=File                  ! File Number.
9860                Array(4,2)=Ujet_ue               ! Jet exit velocity normalized by Uedge.
9870                Array(4,4)=Theta                 ! Tx Side Off Axis Angle.
9880                MAT Array(11:14,1)= Mod          ! Probe volume positions in MODEL coordinates.
9890                MAT Array(11:14,2)= Tun          ! Probe volume positions in TUNNEL coordinates.
9900                MAT Array(11:14,3)= Tcs1         ! Tx side traverse positions in Tcs8 coordinates.
9910                MAT Array(11:14,4)= Tcs2         ! Rx side traverse positions in Tcs8 coordinates.
9920                MAT Array(21,1:3)= Index         ! Index of refraction of for laser light (eg: Nair,Nglass,Nwater).
9930                MAT Array(22:24,1:3)= Theta1     ! Angles between LASER & TUNNEL UVW laser beams in Air (N=Index1).
9940                MAT Array(25:27,1:3)= Theta3     ! Angles between LASER & TUNNEL UVW laser beams in Water (N=Index3).
9950                MAT Array(31,1:3)= Beam_spc      ! Beam spacing at lens.
9960                MAT Array(32,1:3)= Focl_len      ! Focal length.
9970                MAT Array(33,1:3)= Beam_sep      ! Beam separation angle in degrees (full angle).
9980                MAT Array(34,1:3)= Wave_len      ! Wave length.
9990                MAT Array(35,1:3)= Frng_spc      ! Fringe spacing.
10000               MAT Array(36,1:3)= Brg_frq       ! Bragg frequency.
10010               MAT Array(37,1:3)= Mix_frq       ! Mixing frequency.
10020               MAT Array(38,1:3)= Mea_sgn       ! Sign of measured frequency in velocity equation.
10030               MAT Array(39,1:3)= Brg_sgn       ! Sign of bragg    frequency in velocity equation.
10040               MAT Array(40,1:3)= Mix_sgn       ! Sign of mixing   frequency in velocity equation.
10050               MAT Array(41,1:3)= Coin          ! Coincidence criteria
10060               Array(42,1)=Umin                 ! Frequency minimum for U calculation.
10070               Array(42,2)=Vmin                 ! Frequency minimum for V calculation.
10080               Array(42,3)=Wmin                 ! Frequency minimum for W calculation.
10090               Array(43,1)=Umax                 ! Frequency maximum for U calculation.
10100               Array(43,2)=Vmax                 ! Frequency maximum for V calculation.
10110               Array(43,3)=Wmax                 ! Frequency maximum for W calculation.
10120               Array(51,1)=Nreads               ! Number of desired samples.
10130               Array(52,1)=Nsam                 ! Number of acquired samples.
10140               Array(51,2)=Atime                ! Acquisition time.
10150               Array(52,2)=Ctime                ! Coincidence time.
10160               Array(51,3)=At_exp               ! Acquisition time exponent.
10170               Array(52,3)=Ct_exp               ! Coincidence time exponent.
10180               Array(51,4)=Paxis                ! Axis for plots.
10190               Array(52,4)=Clip                 ! Clip:  1 turn histogram clipping on; 0 turns it off.
10200               RETURN
10210 Read:         ! This subroutine extracts (reads) the current tunnel conditions, laser parameters, and histogram
10220               ! & profile scales from the Array(*).
10230               Date=TIMEDATE                    ! Date.
10240               Mach=Array(1,2)                  ! Mach Number.
10250               Alpha(1)=Array(1,4)              ! Angle of Attack.
10260               Time=Date                        ! Time.
10270               Temp=Array(2,2)                  ! Room Temperature (deg. F).
10280               Alpha(2)=Array(2,4)              ! Angle of Yaw.
10290               Uedge=Array(3,2)                 ! Freestream Velocity.
10300               Alpha(3)=Array(3,4)              ! Angle of Roll.
10310               Ujet_ue=Array(4,2)               ! Jet exit velocity normalized by Uedge.
10320               Theta=Array(4,4)                 ! Tx Side Off Axis Angle.
10330               MAT Mod= Array(11:14,1)          ! Probe volume positions in MODEL coordinates.
10340               MAT Tun= Array(11:14,2)          ! Probe volume positions in TUNNEL coordinates.
10350               MAT Tcs1= Array(11:14,3)         ! Tx side traverse positions in Tcs8 coordinates.
10360               MAT Tcs2= Array(11:14,4)         ! Rx side traverse positions in Tcs8 coordinates.
10370               MAT Index= Array(21,1:3)         ! Index of refraction of for laser light (eg: Nair,Nglass,Nwater).
10380               MAT Theta1= Array(22:24,1:3)     ! Angles between LASER & TUNNEL UVW laser beams in Air (N=Index1).
10390               MAT Theta3= Array(25:27,1:3)     ! Angles between LASER & TUNNEL UVW laser beams in Water (N=Index3).
10400               MAT Beam_spc= Array(31,1:3)      ! Beam spacing at lens.
10410               MAT Focl_len= Array(32,1:3)      ! Focal length.
```

```
10420              MAT Beam_sep= Array(33,1:3)      ! Beam separation angle in degrees (full angle).
10430              MAT Wave_len= Array(34,1:3)      ! Wave length.
10440              MAT Frng_spc= Array(35,1:3)      ! Fringe spacing.
10450              MAT Brg_frq= Array(36,1:3)       ! Bragg frequency.
10460              MAT Mix_frq= Array(37,1:3)       ! Mixing frequency.
10470              MAT Mea_sgn= Array(38,1:3)       ! Sign of measured frequency in velocity equation.
10480              MAT Brg_sgn= Array(39,1:3)       ! Sign of bragg    frequency in velocity equation.
10490              MAT Mix_sgn= Array(40,1:3)       ! Sign of mixing   frequency in velocity equation.
10500              MAT Coin= Array(41,1:3)          ! Coincidence criteria.
10510              Umin=Array(42,1)                 ! Frequency minimum for U calculation.
10520              Vmin=Array(42,2)                 ! Frequency minimum for V calculation.
10530              Wmin=Array(42,3)                 ! Frequency minimum for W calculation.
10540              Umax=Array(43,1)                 ! Frequency maximum for U calculation.
10550              Vmax=Array(43,2)                 ! Frequency maximum for V calculation.
10560              Wmax=Array(43,3)                 ! Frequency maximum for W calculation.
10570              Nreads=Array(51,1)               ! Number of desired samples.
10580              Nsam=Array(52,1)                 ! Number of acquired samples.
10590              Atime=Array(51,2)                ! Acquisition time.
10600              Ctime=Array(52,2)                ! Coincidence time.
10610              At_exp=Array(51,3)               ! Acquisition time exponent.
10620              Ct_exp=Array(52,3)               ! Coincidence time exponent.
10630              Paxis=Array(51,4)                ! Axis for plots.
10640              Clip=Array(52,4)                 ! Clip:  1 turn histogram clipping on; 0 turns it off.
10650              RETURN
10660 Calc:        ! This subroutine uses the current tunnel conditions and laser parameters to calculate and/or update other
10670              ! tunnel conditions and laser parameters.
10680              FOR K=1 TO 3
10690                  IF K=2 THEN
10700                      Beam1=Theta+ATN(Beam_spc(K)/2/Focl_len(K))     ! Angles of the off axis beam pair in air.
10710                      Beam2=Theta-ATN(Beam_spc(K)/2/Focl_len(K))
10720                  ELSE
10730                      Beam1=0+ATN(Beam_spc(K)/2/Focl_len(K))         ! Angles of the on axis beam pairs in air.
10740                      Beam2=0-ATN(Beam_spc(K)/2/Focl_len(K))
10750                  END IF
10760                  Beam1=ASN(Index(1)/Index(3)*SIN(Beam1))            ! Angle of the beam pairs in water.
10770                  Beam2=ASN(Index(1)/Index(3)*SIN(Beam2))
10780                  Beam_sep(K)=Beam1-Beam2                            ! Beam pair separation angle.
10790                  Frng_spc(K)=Wave_len(K)/Index(3)/(2*SIN(Beam_sep(K)/2))/1000   ! Fringe spacing in water (um).
10800              NEXT K
10810              MAT Array(33,1:3)= Beam_sep         ! Beam separation angle in degrees (full angle).
10820              MAT Array(35,1:3)= Frng_spc         ! Fringe spacing.
10830              ! Calculate the TCS to TUNNEL (and visa versa) traverse coordinate system transformation matrices.
10840              Fs=Focl_len(1)              ! Focal length of  sending  side lenses (inches).
10850              Fr=Focl_len(1)              ! Focal length of receiving side lenses (inches).
10860              Bs=Beam_spc(1)              ! Beam spacing at sending lenses        (inches).
10870              Br=3                        ! Receiving side lens diameter          (inches).
10880              Ts=Theta                    ! Sending side off axis angle          (degrees).
10890              Tr=17.05                    ! Receiving side off axis angle        (degrees).
10900              !        Ta is the offaxis sending side auxiliary rotation angle (degrees).
10910              CALL Ctm_tcs(Tcs2tun1(*),Tcs2tun2(*),Tun2tcs1(*),Tun2tcs2(*),Fs,Fr,Bs,Br,Index(*),Ts,Tr,Ta)
10920              ! Calculate the LASER to TUNNEL (and visa versa) velocity coordinate system transformation matrices.
10930              CALL Refract(Index(*),Theta1(*),Theta3(*))
10940.             CALL Ctm_ldv(Theta3(*),Tun2ldv(*),Ldv2tun(*))
10950              ! Calculate the TUNNEL to MODEL (and visa versa) coordinate system transformation matrices.
10960              CALL Ctm_mod(Alpha(*),Mod2tun(*),Tun2mod(*))
10970              ! Define the coincidence mask depending on the value of Coin(*).
10980              Cmask=Coin(1)*1+Coin(2)*2+Coin(3)*4
10990              ! Define Paxis$ depending on the value of Paxis.
11000              SELECT Paxis
11010              CASE 1
11020                  Paxis$="X"
11030              CASE 2
11040                  Paxis$="Y"
11050              CASE 3
11060                  Paxis$="Z"
11070              CASE 4
11080                  Paxis$="A"
11090              CASE ELSE
11100                  Paxis=2
11110                  Paxis$="Y"
11120                  GOSUB M3k4
11130              END SELECT
11140              ! If the Run number or File number have not been defined then have the user enter their values.
11150              IF Run=0 OR File=0 THEN
11160                  CALL Enter_value("Run Number ",Run,"3D.2D")
11170                  CALL Enter_value("File Number ",File,"3D")
11180                  GOTO 11150
11190              END IF
11200              RETURN
11210              END
```

```
11220 Do_nothing:     SUB Do_nothing
11230                     ! Description:
11240                     !     This subprogram is called when the keys on the keyboard are pressed during TCS8 traverse
11250                     !     movements. This is done so that any STOP, PAUSE, or RESET keys will be ignored. This prevents
11260                     !     stopping the program while the HP and TCS8 are communicating with each other. Otherwise, they
11270                     !     might get out of sync while communicating resulting in system hang ups.
11280                     ! Variables:
11290                     !     K$          String used to flush the keyboard buffer.
11300                     K$=KBD$
11310                 SUBEND
11320 Menu:           !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
11330 Menu_read:      SUB Menu_read(Menu$(*))
11340                     ! Description:
11350                     !     This subprogram reads in the menu descriptors for each entry of the five menus.
11360                     ! Variables:
11370                     !     Menu      Used as an index to the string array Menu$(*).
11380                     !     Key       Used as an index to the string array Menu$(*).
11390                     !     Menu$(*)  String array where each element describes its corresponding menu subroutine's function.
11400                     !     L$        String use to read in the menu descriptor from the data statements.
11410                     OPTION BASE 1
11420                     DIM L$[80]
11430                     ! Fill all of the menu entry's descriptions with "MxKx".
11440                     FOR Menu=1 TO SIZE(Menu$,1)
11450                         FOR Key=1 TO 8
11460                             Menu$(Menu,Key)="M"&VAL$(Menu)&"K"&VAL$(Key)&":"
11470                         NEXT Key
11480                     NEXT Menu
11490                     ON ERROR GOTO 11570    ! The following while loop will get error#36 when the data statements run out.
11500                     ! For each menu and key, enter the menu entry's description.
11510                     WHILE 1=1
11520                         READ L$
11530                         Menu=VAL(L$[2,2])
11540                         Key=VAL(L$[4,4])
11550                         Menu$(Menu,Key)=L$
11560                     END WHILE
11570                     SUBEXIT
11580                     DATA "M1K1: Laser Alignment"
11590                     DATA        "M2K1: Return to main menu"
11600                     DATA        "M2K2: Sides       : Tx & Rx"
11610                     DATA        "M2K3: Coordinates: MODEL"
11620                     DATA        "M2K4: Mode        : ABSOLUTE"
11630                     DATA        "M2K5: Move X"
11640                     DATA        "M2K6: Move Y"
11650                     DATA        "M2K7: Move Z"
11660                     DATA        "M2K8: Move A"
11670                     DATA "M1K2: Pre Run"
11680                     DATA        "M3K1: Return to MAIN menu"
11690                     DATA        "M3K2: Enter Run & File Numbers"
11700                     DATA        "M3K3: Enter Number of Samples"
11710                     DATA        "M3K4: Select Traverse Axis for Profile"
11720                     DATA        "M3K5: Print Coordinate Transformation Matrices"
11730                     DATA        "M3K6: Setup Graphics"
11740                     DATA        "M3K7: Tunnel Conditions"
11750                     DATA               "M4K1: Return to PRE RUN menu"
11760                     DATA               "M4K2: Load  Tunnel Conditions"
11770                     DATA               "M4K3: Save  Tunnel Conditions"
11780                     DATA               "M4K4: Print Tunnel Conditions"
11790                     DATA               "M4K5: Enter Tunnel Condition Data"
11800                     DATA               "M4K6: Enter Tunnel Condition Names"
11810                     DATA               "M4K7: Enter Tunnel Condition Units"
11820                     DATA               "M4K8: Enter Tunnel Condition Images"
11830                     DATA        "M3K8: Traverse"
11840                     DATA              "M5K1: Return to PRE RUN menu"
11850                     DATA              "M5K2: View & Set TCS8 Positions"
11860                     DATA              "M5K3: View & Set TCS8 Units"
11870                     DATA              "M5K4: View & Set TCS8 Revolution"
11880                     DATA              "M5K5: View & Set TCS8 Velocity"
11890                     DATA              "M5K6: View & Set TCS8 Acceleration"
11900                     DATA "M1K3: Post Run (Dump Graphics)"
11910                     DATA "M1K4: Set Auto Move Positions"
11920                     DATA "M1K5: Move traverse"
11930                     DATA "M1K6: Take data"
11940                     DATA "M1K7: Auto move and take"
11950                     DATA "M1K8: Display Histograms"
11960                 SUBEND
11970 Menu_disp:      SUB Menu_disp(Menu,Menu$(*))
11980                     ! Description:
11990                     !     This subprogram displays the current menu at the top of the CRT.
12000                     ! Variables:
12010                     !     Menu      Used as an index to the string array Menu$(*).
```

```
12020                    !      Key       Used as an index to the string array Menu$(*).
12030                    !      Menu$(*) String array where each element describes its corresponding menu subroutine's function.
12040                    PRINTER IS CRT
12050                    PRINT CHR$(128);            ! Turn off inverse video if it is on.
12060                    IF Menu=0 THEN Menu=1
12070                    FOR Key=1 TO 8
12080                        Menu$(Menu,Key)=Menu$(Menu,Key)&RPT$(" ",50-LEN(Menu$(Menu,Key)))
12090                        PRINT TABXY(1,Key);Menu$(Menu,Key)[3]
12100                    NEXT Key
12110                SUBEND
12120 Menu_status:   SUB Menu_status(Menu,Key,Pen,Menu$(*))
12130                    !  Description:
12140                    !      This subprogram displays the current menu selection in normal or inverse video.  The inverse
12150                    !      video text style indicates that the subroutine for the current menu selection is busy.  The
12160                    !      normal text style indicates that the subroutine for the current menu selection is has completed.
12170                    !  Variables:
12180                    !      Menu      Indicates which of the menus has been selected as the current menu.
12190                    !      Key       Indicates which one of eight menu subroutines in the menu is to be executed.
12200                    !      Pen       Indicates Busy/Ready Status.  Pen=0 for busy.  Pen=1 for ready.
12210                    !      Menu$(*) String array where each element describes its corresponding menu subroutine's function.
12220                    PRINTER IS CRT
12230                    PRINT TABXY(1,Key);CHR$(129-Pen);Menu$(Menu,Key)[3];CHR$(128)
12240                    WAIT .1
12250                SUBEND
12260 Enter:         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
12270 Enter_value:   SUB Enter_value(Name$,Value,Image$)
12280                    !  Description:
12290                    !      This subprogram displays the current value of a variable and then has the user enter its new
12300                    !      value.  The old value will be kept if the RETURN key is pressed and no data is entered.
12310                    !  Variables:
12320                    !      Name$       Name of the variable.
12330                    !      Image$      Image format of the variable.  Used for printing the variable with a format.
12340                    !      Value       Contains the initial value and then the updated value for the variable.
12350                    IF Name$="Date" OR Name$="Time" THEN SUBEXIT
12360                    DISP CHR$(129);
12370                    DISP USING 12380;Name$
12380                    IMAGE #,"Old ",K,"="
12390                    IF Image$<>"" THEN DISP USING "#,"&Image$;Value
12400                    IF Image$="" THEN DISP USING "#,K";Value
12410                    DISP USING 12420;Name$
12420                    IMAGE #,"      Enter new ",K
12430                    INPUT " ? ",Value
12440                    DISP CHR$(128);
12450                SUBEND
12460 Enter_string:  SUB Enter_string(Name$,Value$,Image$)
12470                    !  Description:
12480                    !      This subprogram displays the current value of a string variable and then has the user enter its
12490                    !      new value.  The old value will be kept if the RETURN key is pressed and no data is entered.
12500                    !  Variables:
12510                    !      Name$       Name of the variable.
12520                    !      Value$      Contains the initial value and then the updated value for the string variable.
12530                    DISP CHR$(129);
12540.                   DISP USING 12550;Name$
12550                    IMAGE #,"Old ",K,"="
12560                    DISP USING "#,"&Image$;Value$
12570                    DISP USING 12580;Name$
12580                    IMAGE #,"      Enter new ",K
12590                    INPUT " ? ",Value$
12600                    DISP CHR$(128);
12610                SUBEND
12620 Array:         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
12630 Array_init:    SUB Array_init(Name$(*),Array(*),Image$(*),Units$(*))
12640                    !  Description:
12650                    !      This subprogram reads in default data for each of the variable's names, values, image formats,
12660                    !      and units.  These variables include, but are not limited to, the tunnel conditions, laser
12670                    !      parameters, graph scales, traverse positions, and coordinate system transformation matrices.
12680                    !  Variables:
12690                    !      Array(*)    Array of tunnel conditions, laser parameters, graph scales, etc.
12700                    !      Name$(*)    Names for the variables in Array(*).
12710                    !      Image$(*)   Image formats for the variables in Array(*).
12720                    !      Units$(*)   Units for the variables in Array(*).
12730                    !      X           Used as an index to the above arrays and string arrays.
12740                    !      Y           Used as an index to the above arrays and string arrays.
12750                    !      Before      Number of digits before the decimal point in the image format.
12760                    !      After       Number of digits after  the decimal point in the image format.
12770                    ON ERROR GOTO 12950
12780                    READ Y
12790                    FOR X=1 TO SIZE(Name$,2)
12800                        READ Name$(Y,X),Array(Y,X),Image$(Y,X),Units$(Y,X)
12810                        SELECT Image$(Y,X)
```

```
12820                   CASE "0"
12830                      Image$(Y,X)="9D"
12840                   CASE "1" TO "7"
12850                      After=VAL(Image$(Y,X))
12860                      Before=8-After
12870                      Image$(Y,X)=VAL$(Before)&"D."&VAL$(After)&"D"
12880                   CASE "K"
12890                   CASE "N"
12900                   CASE ELSE
12910                      Image$(Y,X)="9D"
12920                   END SELECT
12930                NEXT X
12940                GOTO 12780
12950                SUBEXIT
12960                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
12970                DATA  1,  Date     ,      0,0,"" ,  Mach     ,      0,4,"" ,  ""       ,      0,0,"" ,  Alpha1   ,    0,4,3
12980                DATA  2,  Time     ,      0,0,"" ,  Temp     , 68.5,4,3F ,  ""       ,      0,0,"" ,  Alpha2   ,    0,4,3
12990                DATA  3,  Run      ,      0,2,"" ,  Uedge    ,.0762,4,m/s,  ""       ,      0,0,"" ,  Alpha3   ,    0,4,3
13000                DATA  4,  File     ,      0,0,"" ,  Ujet/Ue  ,    0,4,m/s,  ""       ,      0,0,"" ,  Theta    ,   45,4,3
13010                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13020                DATA 11,  Xmod     ,      0,4,in ,  Xtun     ,      0,4,in ,  X1tcs    ,      0,4,in ,  X2tcs    ,    0,4,in
13030                DATA 12,  Ymod     ,      0,4,in ,  Ytun     ,      0,4,in ,  Y1tcs    ,      0,4,in ,  Y2tcs    ,    0,4,in
13040                DATA 13,  Zmod     ,      0,4,in ,  Ztun     ,      0,4,in ,  Z1tcs    ,      0,4,in ,  Z2tcs    ,    0,4,in
13050                DATA 14,  Amod     ,      0,4,in ,  Atun     ,      0,4,in ,  A1tcs    ,      0,4,in ,  A2tcs    ,    0,4,in
13060                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13070                DATA 21,  Index1   ,1.000,3,"" ,  Index2   ,1.430,3,"" ,  Index3   ,1.333,3,"" ,  ""       ,    0,0,""
13080                DATA 22,  Theta1AU,      0,4,3  ,  Theta1AV,     90,4,3  ,  Theta1AW,     90,4,3  ,  ""       ,    0,0,""
13090                DATA 23,  Theta1BU,     45,4,3  ,  Theta1BV,    135,4,3  ,  Theta1BW,     90,4,3  ,  ""       ,    0,0,""
13100                DATA 24,  Theta1CU,     90,4,3  ,  Theta1CV,     90,4,3  ,  Theta1CW,      0,4,3  ,  ""       ,    0,0,""
13110                DATA 25,  Theta3AU,      0,4,3  ,  Theta3AV,     90,4,3  ,  Theta3AW,     90,4,3  ,  ""       ,    0,0,""
13120                DATA 26,  Theta3BU,     45,4,3  ,  Theta3BV,    135,4,3  ,  Theta3BW,     90,4,3  ,  ""       ,    0,0,""
13130                DATA 27,  Theta3CU,     90,4,3  ,  Theta3CV,     90,4,3  ,  Theta3CW,      0,4,3  ,  ""       ,    0,0,""
13140                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13150                DATA 31,  UBeamSpc,2.362,3,in ,  VBeamSpc,2.362,3,in ,  WBeamSpc,2.362,3,in ,  ""       ,    0,0,""
13160                DATA 32,  UFoclLen,19.413,3,in,  VFoclLen,19.413,3,in,  WFoclLen,19.413,3,in,  ""       ,    0,0,""
13170                DATA 33,  UBeamSep,0.000,3,3  ,  VBeamSep,0.000,3,3  ,  WBeamSep,0.000,3,3  ,  ""       ,    0,0,""
13180                DATA 34,  UWaveLen,476.5,3,nm ,  VWaveLen,514.5,3,nm ,  WWaveLen,488.0,3,nm ,  ""       ,    0,0,""
13190                DATA 35,  UFrngSpc,00.00,3,um ,  VFrngSpc,00.00,3,um ,  WFrngSpc,00.00,3,um ,  ""       ,    0,0,""
13200                DATA 36,  Ubrag    ,40.00,4,MHz,  Vbrag    ,40.00,4,MHz,  Wbrag    ,40.00,4,MHz,  ""       ,    0,0,""
13210                DATA 37,  Umix     ,39.90,4,MHz,  Vmix     ,39.90,4,MHz,  WMix     ,39.90,4,MHz,  ""       ,    0,0,""
13220                DATA 38,  UmeaSgn ,    +1,0,"" ,  VmeaSgn ,    +1,0,"" ,  WmeaSgn ,    +1,0,"" ,  ""       ,    0,0,""
13230                DATA 39,  UbrgSgn ,    -1,0,"" ,  VbrgSgn ,    -1,0,"" ,  WbrgSgn ,    -1,0,"" ,  ""       ,    0,0,""
13240                DATA 40,  UmixSgn ,    +1,0,"" ,  VmixSgn ,    +1,0,"" ,  WmixSgn ,    +1,0,"" ,  ""       ,    0,0,""
13250                DATA 41,  U coin   ,     1,0,"" ,  V coin   ,     1,0,"" ,  W coin   ,     1,0,"" ,  ""       ,    0,0,""
13260                DATA 42,  UFreqMin,    -99,4,MHz,  VFreqMin,    -99,4,MHz,  WFreqMin,    -99,4,MHz,  ""       ,    0,0,""
13270                DATA 43,  UFreqMax,     99,4,MHz,  VFreqMax,     99,4,MHz,  WFreqMax,     99,4,MHz,  ""       ,    0,0,""
13280                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13290                DATA 51,  Nreads   , 1000,0,"" ,  Atime    ,     30,6,s  ,  ATexp    ,     10,0,"" ,  Paxis    ,    2,0,""
13300                DATA 52,  Nsam     , 1000,0,"" ,  Ctime    , .1E-2,6,s  ,  CTexp    ,      5,0,"" ,  Clip     ,    0,0,""
13310                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13320                DATA 61,  Xmin1    , 0.00,1,"" ,  Xmax1    , 1.00,1,"" ,  Ymin1    ,      0,0,"" ,  Ymax1    ,  100,0,""
13330                DATA 62,  Xmin2    , 0.00,1,"" ,  Xmax2    , 1.00,1,"" ,  Ymin2    ,      0,0,"" ,  Ymax2    ,  100,0,""
13340.               DATA 63,  Xmin3    , 0.00,1,"" ,  Xmax3    , 1.00,1,"" ,  Ymin3    ,      0,0,"" ,  Ymax3    ,  100,0,""
13350                DATA 64,  Xmin4    , 0.00,1,"" ,  Xmax4    , 2.00,1,"" ,  Ymin4    ,      0,0,"" ,  Ymax4    ,  100,0,""
13360                DATA 65,  Xmin5    , 0.00,1,"" ,  Xmax5    , 2.00,1,"" ,  Ymin5    ,      0,0,"" ,  Ymax5    ,  100,0,""
13370                DATA 66,  Xmin6    ,      0,1,"" ,  Xmax6    ,      3,1,"" ,  Ymin6    , -1.5,2,"" ,  Ymax6    ,  1.5,2,""
13380                DATA 67,  Xmin7    ,      0,1,"" ,  Xmax7    ,     .5,1,"" ,  Ymin7    , -1.5,2,"" ,  Ymax7    ,  1.5,2,""
13390                DATA 68,  Xmin8    ,-.025,3,"" ,  Xmax8    , .025,3,"" ,  Ymin8    , -1.5,2,"" ,  Ymax8    ,  1.5,2,""
13400                DATA 69,  Xmin9    ,    -.1,2,"" ,  Xmax9    ,     .1,2,"" ,  Ymin9    , -1.5,2,"" ,  Ymax9    ,  1.5,2,""
13410                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13420                DATA 71,  Xmin1    ,    935,0,pxl,  Xmax1    , 1235,0,pxl,  Ymin1    ,    725,0,pxl,  Ymax1    ,  825,0,pxl
13430                DATA 72,  Xmin2    ,    935,0,pxl,  Xmax2    , 1235,0,pxl,  Ymin2    ,    585,0,pxl,  Ymax2    ,  685,0,pxl
13440                DATA 73,  Xmin3    ,    935,0,pxl,  Xmax3    , 1235,0,pxl,  Ymin3    ,    445,0,pxl,  Ymax3    ,  545,0,pxl
13450                DATA 74,  Xmin4    ,    935,0,pxl,  Xmax4    , 1235,0,pxl,  Ymin4    ,    305,0,pxl,  Ymax4    ,  405,0,pxl
13460                DATA 75,  Xmin5    ,    935,0,pxl,  Xmax5    , 1235,0,pxl,  Ymin5    ,    165,0,pxl,  Ymax5    ,  265,0,pxl
13470                DATA 76,  Xmin6    ,     75,0,pxl,  Xmax6    ,    325,0,pxl,  Ymin6    ,    525,0,pxl,  Ymax6    ,  825,0,pxl
13480                DATA 77,  Xmin7    ,    425,0,pxl,  Xmax7    ,    675,0,pxl,  Ymin7    ,    525,0,pxl,  Ymax7    ,  825,0,pxl
13490                DATA 78,  Xmin8    ,     75,0,pxl,  Xmax8    ,    325,0,pxl,  Ymin8    ,    165,0,pxl,  Ymax8    ,  465,0,pxl
13500                DATA 79,  Xmin9    ,    425,0,pxl,  Xmax9    ,    675,0,pxl,  Ymin9    ,    165,0,pxl,  Ymax9    ,  465,0,pxl
13510                !      Y     *********X=1*********    *********X=2*********    *********X=3*********    *********X=4********
13520                DATA 81,  Xdiv1    ,      5,0,"" ,  Ydiv1    ,      4,0,"" ,  Xdiv6    ,      6,0,"" ,  Ydiv6    ,    6,0,""
13530                DATA 82,  Xdiv2    ,      5,0,"" ,  Ydiv2    ,      4,0,"" ,  Xdiv7    ,      5,0,"" ,  Ydiv7    ,    6,0,""
13540                DATA 83,  Xdiv3    ,      5,0,"" ,  Ydiv3    ,      4,0,"" ,  Xdiv8    ,      2,0,"" ,  Ydiv8    ,    6,0,""
13550                DATA 84,  Xdiv4    ,      5,0,"" ,  Ydiv4    ,      4,0,"" ,  Xdiv9    ,      4,0,"" ,  Ydiv9    ,    6,0,""
13560                DATA 85,  Xdiv5    ,      5,0,"" ,  Ydiv5    ,      4,0,"" ,  ""       ,      0,0,"" ,  ""       ,    0,0,""
13570            SUBEND
13580 Array_print:   SUB Array_print(Array(*),Name$(*),Image$(*),Units$(*))
13590                ! Description:
13600                !     This subprogram prints the values of each of the variables with their names, image formats, and
13610                !     units.  These variables include, but are not limited to, the tunnel conditions, laser
```

```
13620                  !      parameters, and graph scales.
13630                  !  Variables:
13640                  !      Array(*)    Array of tunnel conditions, laser parameters, graph scales, etc.
13650                  !      Name$(*)    Names for the variables in Array(*).
13660                  !      Image$(*)   Image formats for the variables in Array(*).
13670                  !      Units$(*)   Units for the variables in Array(*).
13680                  !      X           Used as in index to the above arrays and string arrays.
13690                  !      Y           Used as in index to the above arrays and string arrays.
13700                  PRINT USING "#,5/"
13710                  FOR Y=1 TO SIZE(Array,1)
13720                      MAT SEARCH Array(Y,*),#LOC(<>0);L1
13730                      MAT SEARCH Name$(Y,*),#LOC(<>"");L2
13740                      IF L1+L2=0 AND L3=0 THEN 13980
13750                      L3=L1+L2
13760                      PRINT USING "#,28X"
13770                      FOR X=1 TO SIZE(Array,2)
13780                          SELECT Name$(Y,X)
13790                          CASE ""
13800                              PRINT USING "#,28X"
13810                          CASE "Date"
13820                              L$=DATE$(Array(Y,X))
13830                              L$=L$[1,2]&L$[4,6]&L$[8,11]
13840                              PRINT USING "#,8A,A,9A,X,3A,6X";TRIM$(Name$(Y,X)),"=",L$,Units$(Y,X)
13850                          CASE "Time"
13860                              L$=" "&TIME$(Array(Y,X))
13870                              PRINT USING "#,8A,A,9A,X,3A,6X";TRIM$(Name$(Y,X)),"=",L$,Units$(Y,X)
13880                          CASE ELSE
13890                              IF Image$(Y,X)="" THEN Image$(Y,X)="9D"
13900                              ON ERROR GOTO 13930
13910                              PRINT USING "#,8A,A,"&Image$(Y,X)&",X,3A,6X";TRIM$(Name$(Y,X)),"=",Array(Y,X),Units$(Y,X)
13920                              GOTO 13950
13930                              OFF ERROR
13940                              PRINT USING "#,8A,A,K,X,3A,6X";TRIM$(Name$(Y,X)),"=",Array(Y,X),Units$(Y,X)
13950                          END SELECT
13960                      NEXT X
13970                      PRINT
13980                  NEXT Y
13990              SUBEND
14000 Change:      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
14010 Change:      SUB Change(Type$,Array(*),Name$(*),Image$(*),Units$(*))
14020                  !  Description:
14030                  !          This subprogram displays on the CRT the values of each of the variables with their names,
14040                  !      image formats, and units.  The user can select one of the variables and enter a new value,
14050                  !      name, image format, or units.  The user selects the particular variable by using the
14060                  !      left, right, up, and down cursor keys.  The selected variable will appear in inverse video.
14070                  !      When it is not selected, it will appear in normal text.  When the user has selected the
14080                  !      appropriate variable he should then press the "Select" key on the keyboard.  Then, depending on
14090                  !      the value of Type$ he will be asked to enter a new value, name, image format, or units.  To
14100                  !      exit the change variables mode press the "Escape" key.
14110                  !          There are three types of data that are passed to the subprogram.  The first type of data
14120                  !      includes, but is not limited to, the tunnel conditions, laser parameters, and graph scales.
14130                  !      With this first type the user is allowed to enter new variable values, names, image formats, and
14140                  !      units.  The second type of data is the "Auto Move and Take" data.  These data are for the pre
14150                  !      programed traverse positions used in a profile scan.  The third type of data is the "View and
14160                  !      Set TCS8 parameters" data acquired from and then sent back to the TCS8.
14170                  !  Variables:
14180                  !      Array(*)    Array whose values, names, image formats, or units are to be modified.
14190                  !      Name$(*)    Names for the variables in Array(*).
14200                  !      Image$(*)   Image formats for the variables in Array(*).
14210                  !      Units$(*)   Units for the variables in Array(*).
14220                  !      Type$       Indicates which type of data is to be entered.
14230                  !                      Type$="VALUES" has the user enter a new value for the selected variable.
14240                  !                      Type$="NAMES"  has the user enter a new name  for the selected variable.
14250                  !                      Type$="IMAGES" has the user enter a new image format for the selected variable.
14260                  !                      Type$="UNITS"  has the user enter a new units for the selected variable.
14270                  !      X           Used as in index to the above arrays and string arrays.
14280                  !      Y           Used as in index to the above arrays and string arrays.
14290                  PRINTER IS CRT
14300                  FOR Y=1 TO SIZE(Array,1)
14310                      FOR Y1=Y TO SIZE(Array,1)
14320                          FOR X=1 TO SIZE(Array,2)
14330                              IF Name$(Y1,X)<>"" THEN 14380
14340                          NEXT X
14350                      NEXT Y1
14360                      CLEAR SCREEN
14370                      SUBEXIT
14380                      FOR Y2=Y1 TO SIZE(Array,1)
14390                          FOR X=1 TO SIZE(Array,2)
14400                              IF Name$(Y2,X)<>"" THEN 14430
14410                          NEXT X
```

```
14420                      GOTO 14440
14430                  NEXT Y2
14440                  FOR Y2=Y2 TO SIZE(Array,1)
14450                      FOR X=1 TO SIZE(Array,2)
14460                          IF Name$(Y2,X)<>"" THEN 14490
14470                      NEXT X
14480                  NEXT Y2
14490                  Y2=Y2-1
14500                  CLEAR SCREEN
14510                  CALL Display(Type$,Y1,Y2,Array(*),Name$(*),Image$(*),Units$(*))
14520                  Done=0
14530                  X=1
14540                  Y=Y1
14550                  ON KBD ALL,15 GOSUB Kbd
14560 Wait:            IF NOT Done THEN Wait
14570                  OFF KBD
14580                  CLEAR SCREEN
14590                  Y=Y2
14600              NEXT Y
14610              SUBEXIT
14620 Kbd:         CALL Update(Type$,X,Y,Y1,Y2,Done,Array(*),Name$(*),Image$(*),Units$(*))
14630              RETURN
14640          SUBEND
14650 Display: SUB Display(Type$,Y1,Y2,Array(*),Name$(*),Image$(*),Units$(*))
14660              !  Description:
14670              !      This subprogram displays on the CRT the values of each of variables with their names, image
14680              !      formats, and units.
14690              !  Variables:
14700              !      Array(*)    Array whose values, names, image formats, or units are to be modified.
14710              !      Name$(*)    Names for the variables in Array(*).
14720              !      Image$(*)   Image formats for the variables in Array(*).
14730              !      Units$(*)   Units for the variables in Array(*).
14740              !      Type$       Indicates which type of data is to be entered.
14750              !                      Type$="VALUES" has the user enter a new value for the selected variable.
14760              !                      Type$="NAMES"  has the user enter a new name  for the selected variable.
14770              !                      Type$="IMAGES" has the user enter a new image format for the selected variable.
14780              !                      Type$="UNITS"  has the user enter a new units for the selected variable.
14790              !      X           Used as in index to the above arrays and string arrays.
14800              !      Y           Used as in index to the above arrays and string arrays.
14810              FOR Y=Y1 TO Y2
14820                  FOR X=1 TO SIZE(Array,2)
14830                      CALL Select(Type$,X,Y,Y1,Y2,0,Array(*),Name$(*),Image$(*),Units$(*))
14840                  NEXT X
14850              NEXT Y
14860              CALL Select(Type$,1,Y1,Y1,Y2,1,Array(*),Name$(*),Image$(*),Units$(*))
14870          SUBEND
14880 Select:  SUB Select(Type$,X,Y,Y1,Y2,C,Array(*),Name$(*),Image$(*),Units$(*))
14890              !  Description:
14900              !      This subprogram displays on the CRT the value of one variable along with its names, image
14910              !      format, and units.
14920              !  Variables:
14930              !      Array(*)    Array whose values, names, image formats, or units are to be modified.
14940              !      Name$(*)    Names for the variables in Array(*)
14950              !      Image$(*)   Image formats for the variables in Array(*)
14960              !      Units$(*)   Units for the variables in Array(*)
14970              !      Type$       Indicates which type of data is to be entered.
14980              !                      Type$="VALUES" has the user enter a new value for the selected variable.
14990              !                      Type$="NAMES"  has the user enter a new name for the selected variable.
15000              !                      Type$="IMAGES" has the user enter a new image format for the selected variable.
15010              !                      Type$="UNITS"  has the user enter a new units for the selected variable.
15020              !      X           Used as in index to the above arrays and string arrays.
15030              !      Y           Used as in index to the above arrays and string arrays.
15040              PRINT CHR$(128+C);TABXY(26*X-24,15+Y-Y1+1);
15050              PRINT RPT$(" ",23);TABXY(26*X-24,15+Y-Y1+1);
15060              IF Name$(Y,X)="" AND Array(Y,X)=0 THEN 15260
15070              Img$=Image$(Y,X)
15080              Unt$=Units$(Y,X)
15090              IF Image$(Y,X)="" THEN Img$="K"
15100              IF Units$(Y,X)="" THEN Unt$="   "
15110              SELECT Type$
15120              CASE "VALUES"
15130                  SELECT Name$(Y,X)
15140                  CASE "Date"
15150                  CASE "Time"
15160                  CASE ELSE
15170                      PRINT USING "#,10A,A,"&Img$&",X,3A";Name$(Y,X),":",Array(Y,X),Unt$
15180                  END SELECT
15190              CASE "NAMES"
15200                  PRINT USING "#,10A,A,8A";Name$(Y,X),":",Name$(Y,X)
15210              CASE "UNITS"
```

```
15220                     PRINT USING "#,10A,A,8A";Name$(Y,X),":",Units$(Y,X)
15230                 CASE "IMAGES"
15240                     PRINT USING "#,10A,A,8A";Name$(Y,X),":",Image$(Y,X)
15250                 END SELECT
15260                 PRINT CHR$(128);
15270             SUBEND
15280 Update:     SUB Update(Type$,X,Y,Y1,Y2,Done,Array(*),Name$(*),Image$(*),Units$(*))
15290                 !  Description:
15300                 !      This subprogram scrolls through the variables displayed on the CRT and has the user enter
15310                 !      updated values.  The user can select one of the variables and enter a new value, name, image
15320                 !      format, or units.  The user selects the particular variable by using the left, right, up, down
15330                 !      cursor keys.  This subprogram will only have been called after a keyboard key has been pressed.
15340                 !      If a cursor key has been pressed then the previously selected variable will be redisplayed in
15350                 !      normal text and the new selected variable will appear in inverse video text.  When the user has
15360                 !      selected the appropriate variable he will have pressed the "Select" key on the keyboard.  Then,
15370                 !      depending on the value of the Type$ he will be asked to enter a new value, name, image format,
15380                 !      or units.  To exit the change variables mode the user will have pressed the "Escape" key.
15390                 !  Variables:
15400                 !      Array(*)    Array of tunnel conditions, laser parameters, graph scales, etc.
15410                 !      Name$(*)    Names for the variables in Array(*).
15420                 !      Image$(*)   Image formats for the variables in Array(*).
15430                 !      Units$(*)   Units for the variables in Array(*).
15440                 !      Type$       Indicates which type of data is to be entered.
15450                 !                         Type$="VALUES" has the user enter a new value for the selected variable.
15460                 !                         Type$="NAMES"  has the user enter a new name  for the selected variable.
15470                 !                         Type$="IMAGES" has the user enter a new image format for the selected variable.
15480                 !                         Type$="UNITS"  has the user enter a new units for the selected variable.
15490                 !      X           Used as in index to the above arrays and string arrays.
15500                 !      Y           Used as in index to the above arrays and string arrays.
15510                 DISABLE
15520                 K$=KBD$
15530                 IF K$="" THEN 15990
15540                 SELECT NUM(K$[1,1])
15550                 CASE 27                                                                     ! ESC
15560                     Done=1
15570                 CASE 255
15580                     CALL Select(Type$,X,Y,Y1,Y2,0,Array(*),Name$(*),Image$(*),Units$(*))
15590                     SELECT NUM(K$[2,2])
15600                     CASE 73,80                                                              ! Break,Stop
15610                         PAUSE
15620                     CASE 124                                                                ! Menu
15630                         Done=1
15640                     CASE 38                                                                 ! Select
15650                         CALL Select(Type$,X,Y,Y1,Y2,1,Array(*),Name$(*),Image$(*),Units$(*))
15660                         SELECT Type$
15670                         CASE "VALUES"
15680                             IF Name$(Y,X)="" THEN CALL Enter_string("Name for "&Name$(Y,X),Name$(Y,X),"K")
15690                             IF Image$(Y,X)="" THEN CALL Enter_string("Image for "&Name$(Y,X),Image$(Y,X),"K")
15700                             CALL Enter_value(Name$(Y,X),Array(Y,X),Image$(Y,X))
15710                         CASE "NAMES"
15720                             CALL Enter_string("Name for "&Name$(Y,X),Name$(Y,X),"K")
15730                         CASE "UNITS"
15740                             CALL Enter_string("Units for "&Name$(Y,X),Units$(Y,X),"K")
15750                         CASE "IMAGES"
15760                             CALL Enter_string("Image for "&Name$(Y,X),Image$(Y,X),"K")
15770                         END SELECT
15780                         CALL Select(Type$,X,Y,Y1,Y2,0,Array(*),Name$(*),Image$(*),Units$(*))
15790                         IF X=SIZE(Array,2) THEN Y=Y+1
15800                         X=X+1
15810                     CASE 60                                                                 ! Left
15820                         X=X-1
15830                     CASE 62                                                                 ! Right
15840                         X=X+1
15850                     CASE 94                                                                 ! Up
15860                         Y=Y-1
15870                     CASE 86                                                                 ! Down
15880                         Y=Y+1
15890                     CASE 92                                                                 ! First
15900                         X=1
15910                         Y=1
15920                     END SELECT
15930                     X=(X-1) MOD SIZE(Array,2)+1
15940                     Y=(Y-Y1+1-1) MOD (Y2-Y1+1)+Y1
15950                     IF X<1 THEN X=SIZE(Array,2)
15960                     IF Y<Y1 THEN Y=Y2
15970                     CALL Select(Type$,X,Y,Y1,Y2,1,Array(*),Name$(*),Image$(*),Units$(*))
15980                 END SELECT
15990                 ENABLE
16000                 SUBEXIT
16010             SUBEND
```

```
16020 Misc:           !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
16030 Convert2words: SUB Convert2words(Real,INTEGER High,Low)
16040                   !  Description:
16050                   !      This subprogram converts a single real precision variable into two 16 bit words.  The initial
16060                   !      real precision variables is converted in to a 32 bit integer and then separated into high and
16070                   !      low 16 bit integers.  The most significant 16 bits will be in the "High" variable while the
16080                   !      least significant 16 bits will be placed the the "Low" variable.  The main purpose of this
16090                   !      subprogram is to provide a means to send a 32 bit integer to the LVDAS over the 16 bit high
16100                   !      speed interface.
16110                   !  Variables:
16120                   !      Real    Initial real precision value for the variable.
16130                   !      Hex$    Hex value of "Real".  String length will be 8 bytes for 32 bits.
16140                   !      High    Most  significant 16 bits of integerized "Real".
16150                   !      Low     Least significant 16 bits of integerized "Real".
16160                   Hex$=DVAL$(Real,16)
16170                   High=IVAL(Hex$[1,4],16)
16180                   Low=IVAL(Hex$[5,8],16)
16190               SUBEND
16200 Error:        SUB Error
16210                   ! Description:
16220                   !     This subprogram will print an error message when ever a program error occurs.  The error message
16230                   !     will be displayed at the top of the CRT and also printed on the printers paper.  Such errors
16240                   !     might occur when data to be printed will not fit in the image formats.  Other errors will also
16250                   !     generate a displayed and printed error message.
16260                   BEEP
16270                   DISP ERRM$
16280                   OUTPUT PRT;ERRM$
16290                   Prt=VAL(SYSTEM$("PRINTER IS"))
16300                   PRINTER IS CRT
16310                   PRINT TABXY(95,1);ERRM$
16320                   PRINTER IS Prt
16330                   ERROR SUBEXIT
16340               SUBEND
16350 Scale:        SUB Scale(G)
16360                   ! Description:
16370                   !     This subprogram selects one of nine histogram or profile plots.  The plot's area of the CRT is
16380                   !     selected and scaled to the appropriate scales.
16390                   OPTION BASE 1
16400                   COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                             Legend$(*)
16410                   VIEWPORT Vwprt(G,1)/10.23,Vwprt(G,2)/10.23,Vwprt(G,3)/10.23,Vwprt(G,4)/10.23
16420                   WINDOW Wndw(G,1),Wndw(G,2),Wndw(G,3),Wndw(G,4)
16430               SUBEND
16440 Table:        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
16450 Table:        SUB Table(Table(*))
16460                   ! Description:
16470                   !     This subprogram is used to create a lookup table array.  The lookup table array facilitates
16480                   !     the rapid conversion of raw encoded Macrodyne data into a usable frequency.  Once the table
16490                   !     has been filled, then the raw Macrodyne data can be used as an index to the table array.
16500                   ! Variables:
16510                   !     Table(*)      Lookup table of frequencies.
16520                   !     Mantissa(*)   The 10 bit mantissa part of the raw Macrodyne data (0..1023).
16530                   !     Fringes       The 1 bit Fringe Count part of the raw Macrodyne data (0:16, 1:8 fringes).
16540                   !     Exponent      The 4 bit Exponent part of the raw Macrodyne data.
16550                   !     Time(*)       An array of measurement times for a given number of Fringes and Exponent.
16560                   !     Freq(*)       An array of measured frequencies for a given number of Fringes and Exponent.
16570                   !     Bin           Used to index Mantissa(*).
16580                   !     Min           Used as a subrange index for Table(*).
16590                   !     Max           Used as a subrange index for Table(*).
16600                   OPTION BASE 1
16610                   REAL Mantissa(0:1023),Time(0:1023),Freq(0:1023)
16620                   ! If the last entry in the table in not zero then the table has already been created.
16630                   IF Table(32766) THEN SUBEXIT
16640                   FOR Bin=0 TO 1023               !  Fill Mantissa array.
16650                       Mantissa(Bin)=Bin
16660                   NEXT Bin
16670                   Mantissa(0)=1
16680                   Min=0
16690                   FOR Fringes=0 TO 1              !  0 indicates 16 fringes while 1 indicates 8 fringes.
16700                       FOR Exponent=0 TO 15
16710                           Max=Min+1023
16720                           IF Max=32767 THEN       !  Maximum size of an array is 32766.
16730                               Max=32766
16740                               REDIM Mantissa(0:1022),Time(0:1022),Freq(0:1022)
16750                           END IF
16760                           DISP Fringes,Exponent
16770                          !MAT Time= Mantissa*(2^(Exponent-1)/500000000)         ! Use this line with new macrodynes.
16780                           MAT Time= Mantissa*(2^(Exponent-3)/500000000)         ! Use this line with old macrodynes.
16790                           MAT Freq= (2^(4-Fringes))/Time
16800                           MAT Freq= Freq/(1000000)
```

```
16810                       MAT Table(Min:Max)= Freq
16820                       Min=Min+1024
16830                   NEXT Exponent
16840               NEXT Fringes
16850           SUBEND
16860 Lvdas:    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
16870 Lvdas_init:    SUB Lvdas_init(@Lvdas)
16880               ! Description:
16890               !         This subprogram is used to initialize the HP98622-66501 Rev B 16-bit General Purpose
16900               !     Input Output (GPIO) interface.  The subprogram also opens the LVDAS path on the HP computer
16910               !     for command and data transfer.  The I/O path is given the name "@Lvdas".  Data transferred
16920               !     from the HP to the LVDAS will use the "OUTPUT @Lvdas" statement.  Data transferred to the HP
16930               !     from LVDAS will use the "ENTER @Lvdas" statement.
16940               !         The I/O path has a select code of 12 and is initialized to perform unformatted word
16950               !     transfers without any end of line designations.  The DIP switches on the HP98622-66501 Rev B
16960               !     printed circuit board need to be set as shown below:
16970               !         DIP switches for INT LVL     :  Bit1=0   Bit0=0
16980               !         DIP switches for Select Code :  Bit4=0   Bit3=1   Bit2=1   Bit1=0   Bit0=0
16990               !         DIP switches for DI15to08 clk:  RDY =1   BSY =0   RD  =1
17000               !         DIP switches for DI07to00 clk:  RDY =1   BSY =0   RD  =1
17010               !         DIP switches for Hndsk Levels:  DOUT=0   DIN =0   HSHK=1   PSTS=0   PFLG=0   PCTL=1
17020               ASSIGN @Lvdas TO 12;WORD,FORMAT OFF,EOL ""
17030               OUTPUT @Lvdas USING "#,AA";"HP"
17040           SUBEND
17050 Lvdas_take:    SUB Lvdas_take(@Lvdas,Atime,Ctime,INTEGER At_exp,Ct_exp,Cmask,Nsam)
17060               ! Description:
17070               !     This subprogram samples the two analog, three digital, and two external trigger channels
17080               !     from the LVDAS.  The HP sends a "CS" to sample the LVDAS data with coincidence.  Following the
17090               !     "CS" the HP sends the LVDAS an additional eight words to specify the acquisition and
17100               !     coincidence times, the interarrival and coincidence time exponents, the coincidence mask, and
17110               !     the number of desired samples.  After the desired number of samples is acquired or the desired
17120               !     acquisition time expires then the LVDAS sends to the HP an updated number of samples (Nsam).
17130               !     The updated Nsam may be less that the original Nsam if the desired acquisition time expires
17140               !     before the desired Nsam samples are realized.
17150               ! Variables:
17160               !     Atime    The maximum desired acquisition time (seconds).
17170               !     Ctime    The maximum desired coincidence time (seconds).
17180               !     At1      The upper word of integer of 10000000*Atime.
17190               !     At2      The lower word of integer of 10000000*Atime.
17200               !     Ct1      The upper word of integer of 10000000*Ctime.
17210               !     Ct2      The lower word of integer of 10000000*Ctime.
17220               !     At_exp   Exponent for interarrival times.
17230               !     Ct_exp   Exponent for coincidence times.
17240               !     Nsam     Number of desired samples.
17250               !     Cmask    Coincidence Mask for U,V,W selection.
17260               !     Raw(*)   Array of raw data acquired LVDAS data.
17270               OPTION BASE 1
17280               COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
17290               INTEGER At1,At2,Ct1,Ct2
17300               DISP "Taking Data"
17310               CALL Convert2words(Atime*10000000,At1,At2)
17320               CALL Convert2words(Ctime*10000000,Ct1,Ct2)
17330.              OUTPUT @Lvdas USING "AA,8(W)";"CS",At1,At2,Ct1,Ct2,At_exp,Ct_exp,Cmask,Nsam
17340               ENTER @Lvdas USING "#,W";Nsam
17350               IF Nsam=0 THEN SUBEXIT
17360               REDIM Raw(1:Nsam,1:10)
17370               ENTER @Lvdas USING "#,W";Raw(*)
17380           SUBEND
17390 Data:     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
17400 Data_reduce:    SUB Data_reduce(INTEGER At_exp,Ct_exp,Nsam)
17410               ! Description:
17420               !         This subprogram separates the ten by Nsam Raw(*) data array into multiple one by Nsam
17430               !     arrays.  The frequency arrays Ui,Vi,Wi are extracted from columns 6,7,8 of the Raw data array.
17440               !     The voltage arrays Ai,Bi are extracted from columns 9,10 of the Raw data array.  The
17450               !     interarrival time array Ii is extracted from columns 1 of the Raw data array.  The coincidence
17460               !     time array Ci is extracted from columns 2 of the Raw data array.  The validation word array
17470               !     Valid is extracted from columns 5 of the Raw data array.  If i'th sample acquired contains
17480               !     valid data, then Valid(i) will be equal to one, and zero otherwise.  All values for the Valid
17490               !     array are initially set to one by the LVDAS.
17500               !         The raw data from arrays Ui,Vi,Wi are converted into frequencies by using their initial
17510               !     values as indexes to the frequency look up table array Table(*).  The raw data from arrays
17520               !     Ai,Bi are converted into voltages by multiplying their initial values by 5 volts over 2^15.
17530               !     The raw data from array Ii are converted into interarrival times by multiplying their initial
17540               !     values by 2^At_exp over 10 to get us.  The raw data from array Ci are converted into
17550               !     coincidence times by multiplying their initial values by 2^Ct_exp over 10 to get us.
17560               ! Variables:
17570               !     Table(*)   Lookup table of frequencies.
17580               !     Raw(*)     Array of raw data acquired LVDAS data.
17590               !     Ui(*)      Array of extracted raw U frequency data.
17600               !     Vi(*)      Array of extracted raw V frequency data.
```

```
17610             !     Wi(*)      Array of extracted raw W frequency data.
17620             !     Ai(*)      Array of extracted raw A voltage data.
17630             !     Bi(*)      Array of extracted raw B voltage data.
17640             !     Ii(*)      Array of extracted raw interarrival time data.
17650             !     Ci(*)      Array of extracted raw coincidence time data.
17660             !     Valid(*)   Array of extracted raw validation words.
17670             !     At_exp     Exponent of interarrival times.
17680             !     Ct_exp     Exponent of coincidence times.
17690             !     Nsam       Number of samples acquired.
17700             OPTION BASE 1
17710             COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
17720             REDIM Ui(Nsam),Vi(Nsam),Wi(Nsam),Ai(Nsam),Bi(Nsam),Ii(Nsam),Ci(Nsam),Valid(Nsam)
17730             DISP "Reducing Data"
17740             MAT Ii= Raw(*,1)
17750             MAT Ci= Raw(*,2)
17760             MAT Valid= Raw(*,5)
17770             MAT Ui= Raw(*,6)
17780             MAT Vi= Raw(*,7)
17790             MAT Wi= Raw(*,8)
17800             MAT Ai= Raw(*,9)
17810             MAT Bi= Raw(*,10)
17820             FOR K=1 TO Nsam
17830                 Ui(K)=Table(Ui(K))
17840                 Vi(K)=Table(Vi(K))
17850                 Wi(K)=Table(Wi(K))
17860             NEXT K
17870             MAT Ai= Ai*(5/32768)
17880             MAT Bi= Bi*(5/32768)
17890             MAT Ii= Ii*(2^At_exp/10)
17900             MAT Ci= Ci*(2^Ct_exp/10)
17910             MAT Ui= Ui . Valid
17920             MAT Vi= Vi . Valid
17930             MAT Wi= Wi . Valid
17940             MAT Ai= Ai . Valid
17950             MAT Bi= Bi . Valid
17960             MAT Ii= Ii . Valid
17970             MAT Ci= Ci . Valid
17980         SUBEND
17990 Data_clip: SUB Data_clip(INTEGER Nsam,REAL Umin,Umax,Vmin,Vmax,Wmin,Wmax)
18000             !   Description:
18010             !       This subprogram compares each of the instantaneous U,V, and W frequencies with user
18020             !       selectable minimum and maximum frequencies.  If the instantaneous value is less than the
18030             !       desired minimum, then the validation word is set to zero.  Also, if the instantaneous value is
18040             !       greater than the desired maximum, then the validation word is set to zero.  The setting of the
18050             !       validation words to zero will have the net effect of discarding the data samples from the data
18060             !       set.  In other words, the data is weighted as zero for the average, sdv, normal and shear
18070             !       stress calculations.
18080             !   Variables:
18090             !       Nsam       Number of samples acquired.
18100             !       Ui(*)      Array of instantaneous U frequencies (MHz).
18110             !       Vi(*)      Array of instantaneous V frequencies (MHz).
18120             !       Wi(*)      Array of instantaneous W frequencies (MHz).
18130             !       Valid(*)   Array of sample validation words.
18140             !       Umin       The minimum acceptable U frequency (MHz).
18150             !       Umax       The maximum acceptable U frequency (MHz).
18160             !       Vmin       The minimum acceptable V frequency (MHz).
18170             !       Vmax       The maximum acceptable V frequency (MHz).
18180             !       Wmin       The minimum acceptable W frequency (MHz).
18190             !       Wmax       The maximum acceptable W frequency (MHz).
18200             OPTION BASE 1
18210             COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
18220             DISP "Clipping Histograms"
18230             FOR K=1 TO Nsam
18240                 MAT SEARCH Ui(*),LOC(<Umin);L,K
18250                 IF L<Nsam THEN Valid(L)=0
18260                 K=L
18270             NEXT K
18280             FOR K=1 TO Nsam
18290                 MAT SEARCH Ui(*),LOC(>Umax);L,K
18300                 IF L<Nsam THEN Valid(L)=0
18310                 K=L
18320             NEXT K
18330             FOR K=1 TO Nsam
18340                 MAT SEARCH Vi(*),LOC(<Vmin);L,K
18350                 IF L<Nsam THEN Valid(L)=0
18360                 K=L
18370             NEXT K
18380             FOR K=1 TO Nsam
18390                 MAT SEARCH Vi(*),LOC(>Vmax);L,K
18400                 IF L<Nsam THEN Valid(L)=0
```

```
18410                       K=L
18420                   NEXT K
18430                   FOR K=1 TO Nsam
18440                       MAT SEARCH Wi(*),LOC(<Wmin);L,K
18450                       IF L<Nsam THEN Valid(L)=0
18460                       K=L
18470                   NEXT K
18480                   FOR K=1 TO Nsam
18490                       MAT SEARCH Wi(*),LOC(>Wmax);L,K
18500                       IF L<Nsam THEN Valid(L)=0
18510                       K=L
18520                   NEXT K
18530                   MAT Ui= Ui . Valid
18540                   MAT Vi= Vi . Valid
18550                   MAT Wi= Wi . Valid
18560                   MAT Ai= Ai . Valid
18570                   MAT Bi= Bi . Valid
18580                   MAT Ii= Ii . Valid
18590                   MAT Ci= Ci . Valid
18600               SUBEND
18610 Data_fconvert: SUB Data_fconvert(Array(*))
18620                   !  Description:
18630                   !      This subprogram takes the frequency values from the arrays Ui,Vi,Wi and replaces them with
18640                   !      velocities after doing the frequency to velocity conversion.
18650                   !  Variables:
18660                   !      Array(*)     An array containing relevant LDV laser and tunnel condition parameters
18670                   !      Frng_spc(*   Fringe Spacings extracted from Array(*).
18680                   !      Brg_frq(*)   Bragg Frequencies extracted from Array(*).
18690                   !      Mix_frq(*)   Mixing Freqs. extracted from Array(*).
18700                   !      Mea_sgn(*)   Measured Freq's. Signs extracted from Array(*)
18710                   !      Brg_sgn(*)   Bragg Freq's. Signs extracted from Array(*).
18720                   !      Mix_sgn(*)   Mixing Freq's. Signs extracted from Array(*).
18730                   !      Ui(*)        Array of instantaneous U data.
18740                   !      Vi(*)        Array of instantaneous V data.
18750                   !      Wi(*)        Array of instantaneous W data.
18760                   !  Equations:
18770                   !      The following equations are used to convert the frequencies to velocities
18780                   !          Velocity = Fs * Ftotal
18790                   !          Ftotal = MeaSgn*Fmeas+BrgSgn*Fbrag+MixSgn*Fmix
18800                   OPTION BASE 1
18810                   COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
18820                   DIM Frng_spc(3),Brg_frq(3),Mix_frq(3),Mea_sgn(3),Brg_sgn(3),Mix_sgn(3)
18830                   DISP "Converting Data"
18840                   MAT Frng_spc= Array(35,1:3)
18850                   MAT Brg_frq= Array(36,1:3)
18860                   MAT Mix_frq= Array(37,1:3)
18870                   MAT Mea_sgn= Array(38,1:3)
18880                   MAT Brg_sgn= Array(39,1:3)
18890                   MAT Mix_sgn= Array(40,1:3)
18900                   MAT Ui= Ui*(Mea_sgn(1))
18910                   MAT Vi= Vi*(Mea_sgn(2))
18920                   MAT Wi= Wi*(Mea_sgn(3))
18930.                  MAT Ui= Ui+(Brg_sgn(1)*Brg_frq(1)+Mix_sgn(1)*Mix_frq(1))
18940                   MAT Vi= Vi+(Brg_sgn(2)*Brg_frq(2)+Mix_sgn(2)*Mix_frq(2))
18950                   MAT Wi= Wi+(Brg_sgn(3)*Brg_frq(3)+Mix_sgn(3)*Mix_frq(3))
18960                   MAT Ui= Ui*(Frng_spc(1))
18970                   MAT Vi= Vi*(Frng_spc(2))
18980                   MAT Wi= Wi*(Frng_spc(3))
18990               SUBEND
19000 Data_sum:     SUB Data_sum(Sum(*),INTEGER N(*),Nsam)
19010                   !  Description:
19020                   !      This subprogram performs the summations on the instantaneous LDV and analog data.  Data
19030                   !      will be weighted as zero in the summations if the value of the validation word is set to zero.
19040                   !      Intermediate arrays will be made so that summations of the products of the LDV and analog data
19050                   !      can be determined.
19060                   !  Variables:
19070                   !      Nsam      Number of samples acquired.
19080                   !      Valid(*)  Array of sample validation words.
19090                   !      Ui(*)     Array of instantaneous U frequency or velocity samples.
19100                   !      Vi(*)     Array of instantaneous V frequency or velocity samples.
19110                   !      Wi(*)     Array of instantaneous W frequency or velocity samples.
19120                   !      Ai(*)     Array of instantaneous A voltage samples.
19130                   !      Bi(*)     Array of instantaneous B voltage samples.
19140                   !      Ii(*)     Array of interarrival times.
19150                   !      Ci(*)     Array of coincidence times.
19160                   !      Uu(*)     Instantaneous product of the instantaneous Ui & Ui.
19170                   !      Vv(*)     Instantaneous product of the instantaneous Vi & Vi.
19180                   !      Ww(*)     Instantaneous product of the instantaneous Wi & Wi.
19190                   !      Aa(*)     Instantaneous product of the instantaneous Ai & Ai.
19200                   !      Bb(*)     Instantaneous product of the instantaneous Bi & Bi.
```

```
19210              !       I2(*)      Instantaneous product of the instantaneous Ii & Ii.
19220              !       C2(*)      Instantaneous product of the instantaneous Ci & Ci.
19230              !       Uv(*)      Instantaneous product of the instantaneous Ui & Vi.
19240              !       Vw(*)      Instantaneous product of the instantaneous Vi & Wi.
19250              !       Wu(*)      Instantaneous product of the instantaneous Wi & Ui.
19260              !       Ab(*)      Instantaneous product of the instantaneous Ai & Bi.
19270              !       Ua(*)      Instantaneous product of the instantaneous Ui & Ai.
19280              !       Va(*)      Instantaneous product of the instantaneous Vi & Ai.
19290              !       Wa(*)      Instantaneous product of the instantaneous Wi & Ai.
19300              !       Sum(1,1)   Summation of the array Ui.
19310              !       Sum(2,1)   Summation of the array Vi.
19320              !       Sum(3,1)   Summation of the array Wi.
19330              !       Sum(4,1)   Summation of the array Ai.
19340              !       Sum(5,1)   Summation of the array Bi.
19350              !       Sum(6,1)   Summation of the array Ii.
19360              !       Sum(7,1)   Summation of the array Ci.
19370              !       Sum(1,2)   Summation of the array Uu.
19380              !       Sum(2,2)   Summation of the array Vv.
19390              !       Sum(3,2)   Summation of the array Ww.
19400              !       Sum(4,2)   Summation of the array Aa.
19410              !       Sum(5,2)   Summation of the array Bb.
19420              !       Sum(6,2)   Summation of the array I2.
19430              !       Sum(7,2)   Summation of the array C2.
19440              !       Sum(1,3)   Summation of the array Uv.
19450              !       Sum(2,3)   Summation of the array Vw.
19460              !       Sum(3,3)   Summation of the array Wu.
19470              !       Sum(4,3)   Summation of the array Ab.
19480              !       Sum(5,3)   Summation of the array Ua.
19490              !       Sum(6,3)   Summation of the array Va.
19500              !       Sum(7,3)   Summation of the array Wa.
19510              OPTION BASE 1
19520              COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
19530              REAL Uu(1000),Vv(1000),Ww(1000),Aa(1000),Bb(1000),I2(1000),C2(1000)
19540              REAL Uv(1000),Vw(1000),Wu(1000),Ab(1000),Ua(1000),Va(1000),Wa(1000)
19550              REDIM Uu(Nsam),Vv(Nsam),Ww(Nsam),Aa(Nsam),Bb(Nsam),I2(Nsam),C2(Nsam)
19560              REDIM Uv(Nsam),Vw(Nsam),Wu(Nsam),Ab(Nsam),Ua(Nsam),Va(Nsam),Wa(Nsam)
19570              DISP "Summing Data"
19580              !
19590              MAT Uu= Ui . Ui
19600              MAT Vv= Vi . Vi
19610              MAT Ww= Wi . Wi
19620              MAT Aa= Ai . Ai
19630              MAT Bb= Bi . Bi
19640              MAT Uv= Ui . Vi
19650              MAT Vw= Vi . Wi
19660              MAT Wu= Wi . Ui
19670              MAT Ab= Ai . Bi
19680              MAT Ua= Ui . Ai
19690              MAT Va= Vi . Ai
19700              MAT Wa= Wi . Ai
19710              MAT I2= Ii . Ii
19720              MAT C2= Ci . Ci
19730              !
19740              Sum(1,1)=SUM(Ui)
19750              Sum(2,1)=SUM(Vi)
19760              Sum(3,1)=SUM(Wi)
19770              Sum(4,1)=SUM(Ai)
19780              Sum(5,1)=SUM(Bi)
19790              Sum(6,1)=SUM(Ii)
19800              Sum(7,1)=SUM(Ci)
19810              Sum(1,2)=SUM(Uu)
19820              Sum(2,2)=SUM(Vv)
19830              Sum(3,2)=SUM(Ww)
19840              Sum(4,2)=SUM(Aa)
19850              Sum(5,2)=SUM(Bb)
19860              Sum(6,2)=SUM(I2)
19870              Sum(7,2)=SUM(C2)
19880              Sum(1,3)=SUM(Uv)
19890              Sum(2,3)=SUM(Vw)
19900              Sum(3,3)=SUM(Wu)
19910              Sum(4,3)=SUM(Ab)
19920              Sum(5,3)=SUM(Ua)
19930              Sum(6,3)=SUM(Va)
19940              Sum(7,3)=SUM(Wa)
19950              MAT N= (SUM(Valid))
19960           SUBEND
19970 Data_calc:    SUB Data_calc(INTEGER N(*),REAL Sum(*),U,V,W,A,B,I,C,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                          U1a1,V1a1,W1a1)
19980              !  Description:
19990              !       This subprogram uses the summations on the instantaneous LDV and analog data as well as the
```

```
20000         !     summations of the products of the LDV and analog data.  The subprogram takes these summations
20010         !     and calculates the averages, standard deviations, and shear stresses.
20020         !  Variables:
20030         !     Nsam      The number of valid samples.
20040         !     Sum(1,1)  Summation of the array Ui.
20050         !     Sum(2,1)  Summation of the array Vi.
20060         !     Sum(3,1)  Summation of the array Wi.
20070         !     Sum(4,1)  Summation of the array Ai.
20080         !     Sum(5,1)  Summation of the array Bi.
20090         !     Sum(6,1)  Summation of the array Ii.
20100         !     Sum(7,1)  Summation of the array Ci.
20110         !     Sum(1,2)  Summation of the array Uu.
20120         !     Sum(2,2)  Summation of the array Vv.
20130         !     Sum(3,2)  Summation of the array Ww.
20140         !     Sum(4,2)  Summation of the array Aa.
20150         !     Sum(5,2)  Summation of the array Bb.
20160         !     Sum(6,2)  Summation of the array I2.
20170         !     Sum(7,2)  Summation of the array C2.
20180         !     Sum(1,3)  Summation of the array Uv.
20190         !     Sum(2,3)  Summation of the array Vw.
20200         !     Sum(3,3)  Summation of the array Wu.
20210         !     Sum(4,3)  Summation of the array Ab.
20220         !     Sum(5,3)  Summation of the array Ua.
20230         !     Sum(6,3)  Summation of the array Va.
20240         !     Sum(7,3)  Summation of the array Wa.
20250         !     U         Average U frequency or velocity.
20260         !     V         Average V frequency or velocity.
20270         !     W         Average W frequency or velocity.
20280         !     A         Average A voltage.
20290         !     B         Average B voltage.
20300         !     I         Average interarrival time.
20310         !     C         Average coincidence time.
20320         !     U1        Standard deviation for U frequency or velocity.
20330         !     V1        Standard deviation for V frequency or velocity.
20340         !     W1        Standard deviation for W frequency or velocity.
20350         !     A1        Standard deviation for A voltage.
20360         !     B1        Standard deviation for B voltage.
20370         !     I1        Standard deviation for interarrival time.
20380         !     C1        Standard deviation for coincidence time.
20390         !     U1v1      Velocity:Velocity Shear Stress.
20400         !     V1w1      Velocity:Velocity Shear Stress.
20410         !     W1u1      Velocity:Velocity Shear Stress.
20420         !     A1b1      Voltage :Voltage  Shear Stress.
20430         !     U1a1      Velocity:Voltage  Shear Stress.
20440         !     V1a1      Velocity:Voltage  Shear Stress.
20450         !     W1a1      Velocity:Voltage  Shear Stress.
20460         DISP "Calculating Results"
20470         Nsam=N(1,1)
20480         IF Nsam>0 THEN
20490             U=Sum(1,1)/Nsam
20500             V=Sum(2,1)/Nsam
20510             W=Sum(3,1)/Nsam
20520             A=Sum(4,1)/Nsam
20530             B=Sum(5,1)/Nsam
20540             I=Sum(6,1)/Nsam
20550             C=Sum(7,1)/Nsam
20560             U1=SQR(ABS(Sum(1,2)/Nsam-U*U))
20570             V1=SQR(ABS(Sum(2,2)/Nsam-V*V))
20580             W1=SQR(ABS(Sum(3,2)/Nsam-W*W))
20590             A1=SQR(ABS(Sum(4,2)/Nsam-A*A))
20600             B1=SQR(ABS(Sum(5,2)/Nsam-B*B))
20610             I1=SQR(ABS(Sum(6,2)/Nsam-I*I))
20620             C1=SQR(ABS(Sum(7,2)/Nsam-C*C))
20630             U1v1=Sum(1,3)/Nsam-U*V
20640             V1w1=Sum(2,3)/Nsam-V*W
20650             W1u1=Sum(3,3)/Nsam-W*U
20660             A1b1=Sum(4,3)/Nsam-A*B
20670             U1a1=Sum(5,3)/Nsam-U*A
20680             V1a1=Sum(6,3)/Nsam-V*A
20690             W1a1=Sum(7,3)/Nsam-W*A
20700         ELSE
20710             U=0
20720             V=0
20730             W=0
20740             A=0
20750             B=0
20760             I=0
20770             C=0
20780             U1=0
20790             V1=0
```

```
20800                         W1=0
20810                         A1=0
20820                         B1=0
20830                         I1=0
20840                         C1=0
20850                         U1v1=0
20860                         V1w1=0
20870                         W1u1=0
20880                         A1b1=0
20890                         U1a1=0
20900                         V1a1=0
20910                         W1a1=0
20920                       END IF
20930                     SUBEND
20940 Data_trnsfrm:       SUB Data_trnsfrm(REAL K(*),U,V,W,U1,V1,W1,U1v1,V1w1,W1u1,U1a1,V1a1,W1a1)
20950                       ! Description:
20960                       !     This subprogram performs a coordinate system transformation on the averages, standard
20970                       !     deviations, and shear stresses.  The coordinate system transformation to be applied is passed
20980                       !     through the "K3X3" array.  If a LASER to TUNNEL coordinate system transformation is to be
20990                       !     performed, then the array "Ldv2tun" array will be passed to the "K3X3" array.  If a TUNNEL to
21000                       !     MODEL coordinate system transformation is to be performed, then the array "Tun2mod" array will
21010                       !     be passed to the "K3X3" array.
21020                       ! Variables:
21030                       !     U         Average U velocity.
21040                       !     V         Average V velocity.
21050                       !     W         Average W velocity.
21060                       !     U1        Standard deviation for U velocity.
21070                       !     V1        Standard deviation for V velocity.
21080                       !     W1        Standard deviation for W velocity.
21090                       !     U1u1      Velocity:Velocity Normal Stress.
21100                       !     U1v1      Velocity:Velocity Shear  Stress.
21110                       !     U1w1      Velocity:Velocity Shear  Stress.
21120                       !     V1u1      Velocity:Velocity Shear  Stress.
21130                       !     V1v1      Velocity:Velocity Normal Stress.
21140                       !     V1w1      Velocity:Velocity Shear  Stress.
21150                       !     W1u1      Velocity:Velocity Shear  Stress.
21160                       !     W1v1      Velocity:Velocity Shear  Stress.
21170                       !     W1w1      Velocity:Velocity Normal Stress.
21180                       !     U1a1      Velocity:Voltage  Shear  Stress.
21190                       !     V1a1      Velocity:Voltage  Shear  Stress.
21200                       !     W1a1      Velocity:Voltage  Shear  Stress.
21210                       !     R(*)      Original U,V,W.
21220                       !     F(*)      Original U1a1,V1a1,W1a1.
21230                       !     P(*)      Original stress terms U1u1,U1v1,...,W1w1.
21240                       !     K3X3      Coordinate system transformation matrix for average and Velocity:Voltage shear stress
21250                       !               conversions.
21260                       !     K9X9      Coordinate system transformation matrix for Velocity:Velocity normal and shear stress
21270                       !               conversions.
21280                       !     S(*)      Transformed U,V,W.
21290                       !     H(*)      Transformed U1a1,V1a1,W1a1.
21300                       !     Q(*)      Transformed stress terms U1u1,U1v1,...,W1w1.
21310                       OPTION BASE 1
21320                       REAL R(3),S(3),F(3),H(3),P(9),Q(9),K3x3(3,3),K9x9(9,9)
21330                       DISP "Transforming Results"
21340                       ! Calculate U1u1,V1v1,W1w1 using U1,V1,W1.
21350                       U1u1=U1*U1
21360                       V1v1=V1*V1
21370                       W1w1=W1*W1
21380                       ! Set U1w1,V1u1,W1v1 equal to W1u1,U1v1,V1w1.
21390                       U1w1=W1u1
21400                       V1u1=U1v1
21410                       W1v1=V1w1
21420                       ! Fill the matrix R with U,V,W.
21430                       R(1)=U
21440                       R(2)=V
21450                       R(3)=W
21460                       ! Fill the matrix F with U1a1,V1a1,W1a1.
21470                       F(1)=U1a1
21480                       F(2)=V1a1
21490                       F(3)=W1a1
21500                       ! Fill the matrix P with U1u1,U1v1,U1w1,V1u1,V1v1,V1w1,W1u1,W1v1,W1w1.
21510                       P(1)=U1u1
21520                       P(2)=U1v1
21530                       P(3)=U1w1
21540                       P(4)=V1u1
21550                       P(5)=V1v1
21560                       P(6)=V1w1
21570                       P(7)=W1u1
21580                       P(8)=W1v1
21590                       P(9)=W1w1
```

```
21600                  ! Define the matrix K9x9 using products of the elements from then matrix K3x3.
21610                  FOR X=1 TO 9
21620                     FOR Y=1 TO 9
21630                        Y1=((Y-1) DIV 3)+1
21640                        X1=((X-1) DIV 3)+1
21650                        Y2=((Y-1) MOD 3)+1
21660                        X2=((X-1) MOD 3)+1
21670                        K9x9(Y,X)=K3x3(Y1,X1)*K3x3(Y2,X2)
21680                     NEXT Y
21690                  NEXT X
21700                  ! Transform matrix R to S using K3x3.
21710                  MAT S= K3x3*R
21720                  ! Transform matrix F to H using K3x3.
21730                  MAT H= K3x3*F
21740                  ! Transform matrix P to Q using K9x9.
21750                  MAT Q= K9x9*P
21760                  ! Extract the transformed U,V,W from the matrix S.
21770                  U=S(1)
21780                  V=S(2)
21790                  W=S(3)
21800                  ! Extract the transformed U1a1,V1a1,W1a1 from the matrix H.
21810                  U1a1=H(1)
21820                  V1a1=H(2)
21830                  W1a1=H(3)
21840                  ! Extract the transformed U1u1,U1v1,U1w1,V1u1,V1v1,V1w1,W1u1,W1v1,W1w1 from the matrix Q.
21850                  U1u1=Q(1)
21860                  U1v1=Q(2)
21870                  U1w1=Q(3)
21880                  V1u1=Q(4)
21890                  V1v1=Q(5)
21900                  V1w1=Q(6)
21910                  W1u1=Q(7)
21920                  W1v1=Q(8)
21930                  W1w1=Q(9)
21940                  ! Calculate U1,V1,W1 using U1u1,V1v1,W1w1.
21950                  U1=SQR(ABS(U1u1))
21960                  V1=SQR(ABS(V1v1))
21970                  W1=SQR(ABS(W1w1))
21980                  ! Return transformed U,V,W,U1,V1,W1,U1v1,V1w1,W1u1,U1a1,V1a1,W1a1 to main program.
21990               SUBEND
22000 Print:        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
22010 Data_print:   SUB Data_print(Axis,Pos,INTEGER Nsam,C$,REAL U,V,W,A,B,I,C,U1,V1,W1,A1,B1,I1,C1,U1v1,V1w1,W1u1,A1b1,
                               U1a1,V1a1,W1a1,Uedge)
22020                  !  Description:
22030                  !      This subprogram prints the averages, standard deviations, and shear & normal stress in
22040                  !      tabular form.  This subprogram may be called several times.  The first call might print the
22050                  !      reduced velocity data when their units are in frequency (MHz).  Subsequent calls will print the
22060                  !      reduced data when their units are in velocity (m/s).  These subsequent calls will print the
22070                  !      the data in one of three coordinate systems: LASER, TUNNEL, and MODEL.
22080                  !  Variables:
22090                  !      U         Average U velocity.
22100                  !      V         Average V velocity.
22110                  !      W         Average W velocity.
22120                  !      A         Average W velocity.
22130                  !      B         Average B voltage.
22140                  !      I         Average interarrival time.
22150                  !      C         Average coincidence  time.
22160                  !      U1        Standard deviation for U velocity.
22170                  !      V1        Standard deviation for V velocity.
22180                  !      W1        Standard deviation for W velocity.
22190                  !      A1        Standard deviation for A voltage.
22200                  !      B1        Standard deviation for B voltage.
22210                  !      I1        Standard deviation for interarrival time.
22220                  !      C1        Standard deviation for coincidence  time.
22230                  !      U1v1      Velocity:Velocity Shear  Stress.
22240                  !      V1w1      Velocity:Velocity Shear  Stress.
22250                  !      W1u1      Velocity:Velocity Shear  Stress.
22260                  !      A1b1      Voltage :Voltage  Shear  Stress.
22270                  !      U1a1      Velocity:Voltage  Shear  Stress.
22280                  !      V1a1      Velocity:Voltage  Shear  Stress.
22290                  !      W1a1      Velocity:Voltage  Shear  Stress.
22300                  !      Axis$     Indicates one of the three axes X,Y,Z being traversed.
22310                  !      Pos       Current Traverse Position.
22320                  !      Nsam      Number of samples acquired.
22330                  !      C$        Indicates units and/or coordinate system of data printed.
22340                  DISP "Printing Results"
22350                  ON ERROR CALL Error
22360                  PRINTER IS PRT;WIDTH 144
22370                  Axis$=CHR$(NUM("X")+Axis-1)
22380                  PRINT USING 22490;L$,Pos,U,U1,U1v1
```

```
22390                PRINT USING 22530;A,A1,A1b1,U1a1
22400                PRINT USING 22500;"N",Nsam,V,V1,V1w1
22410                PRINT USING 22540;B,B1,I1,V1a1
22420                PRINT USING 22510;C$[1,3],W,W1,W1u1
22430                PRINT USING 22550;C,I,C1,W1a1
22440                PRINT USING 22520;Uedge,U1/Uedge,V1/Uedge
22450                PRINT USING 22560;W1/Uedge,(U1v1)/Uedge^2,(V1w1)/Uedge^2,(W1u1)/Uedge^2
22460                IF C$<>"MOD" THEN PRINT
22470                PRINTER IS CRT
22480                OFF ERROR
22490                IMAGE #,8X, A,"=",3D.4D,"   U=",3D.5D,"   U1=",3D.5D,"   U1v1=",4D.6D
22500                IMAGE #,8X, A,"=",   8D,"   V=",3D.5D,"   V1=",3D.5D,"   V1w1=",4D.6D
22510                IMAGE #,8X,3A,       7X,"   W=",3D.5D,"   W1=",3D.5D,"   W1u1=",4D.6D
22520                IMAGE #,18X,          "  UE=",3D.4D," U1/UE=",3D.5D," V1/UE=",3D.4D
22530                IMAGE    "   A =",3D.5D,"   A1 =",3D.5D,"  W/UE =",3D.5D,"   U1a1=",2D.6D
22540                IMAGE    " U/UE=",3D.5D," V/UE =",3D.5D,"   IAT1=",9D,"   V1a1=",2D.6D
22550                IMAGE    "   CT=",9D,"   IAT=",9D,"   CT1 =",9D,"   W1a1=",2D.6D
22560                IMAGE    "   W1/UE=",1D.4D," U1v1/UE2=",2D.4D," V1w1/UE2=",2D.4D," W1u1/UE2=",2D.4D
22570            SUBEND
22580 Data_plot: SUB Data_plot(Array(*),Symbols(*),G,Y,P1,P2,P3,Scale,INTEGER N1,N2,N3)
22590                ! Description:
22600                !     This subprogram plots the averages, standard deviations, or shear & normal stress in
22610                !     the 4 profile plots on the CRT.  This subprogram will typically be called 4 times.  The first
22620                !     call will plot the average velocities normalized by Uedge.  The second call will plot the
22630                !     normalized standard deviations of the velocities.  The third call will plot the normalized
22640                !     velocity shear stresses.  The forth and last call will plot the average and standard deviations
22650                !     of the analog data in the forth and last plot.  Data points outside the plot boundaries will
22660                !     be plotted at the plot boundary.
22670                ! Variables:
22680                !     Array(*)     Array containing the plot positions and scales.
22690                !     Symbols(*)   Array of Symbol arrays.  Each symbol array contains a distinct geometric symbol.
22700                !     G            Indicates which plot that the normalized P1,P2,P3 will be plotted against Y in.
22710                !     Y            Vertical position of the normalized data points in the plot.
22720                !     P1           Horizontal position of the 1st data point (P1 will be normalized by Scale).
22730                !     P2           Horizontal position of the 2nd data point (P2 will be normalized by Scale).
22740                !     P3           Horizontal position of the 3rd data point (P3 will be normalized by Scale).
22750                !     Scale        The value that the horizontal positions will be normalized by.
22760                !     N1           The number of samples contributing to P1's value.  P1 will be plotted if N1>0.
22770                !     N2           The number of samples contributing to P2's value.  P2 will be plotted if N2>0.
22780                !     N3           The number of samples contributing to P3's value.  P3 will be plotted if N3>0.
22790                !     Wndw(*)      Array containing the plot's scales.
22800                !     Vwprt(*)     Array containing the plot's CRT position.
22810                !     Symbol(*)    Array containing a distinct geometric symbol.
22820                OPTION BASE 1
22830                DIM Wndw(4),Vwprt(4),Symbol(20,3)
22840                DISP "Plotting Results"
22850                AREA PEN -1
22860                PEN 1
22870                MAT Wndw= Array(60+G,*)
22880                MAT Vwprt= Array(70+G,*)
22890                VIEWPORT Vwprt(1)/10.23,Vwprt(2)/10.23,Vwprt(3)/10.23,Vwprt(4)/10.23
22900                WINDOW Wndw(1),Wndw(2),Wndw(3),Wndw(4)
22910                CLIP ON
22920                FOR I=0 TO 2
22930                    IF I=0 AND N1=0 THEN 23120
22940                    IF I=1 AND N2=0 THEN 23120
22950                    IF I=2 AND N3=0 THEN 23120
22960                    Sy=I+1
22970                    Noc=Symbols(Sy,0,1)
22980                    REDIM Symbol(Noc,3)
22990                    MAT Symbol= Symbols(Sy,1:Noc,*)
23000                    SELECT I
23010                    CASE 0
23020                        X=P1*Scale
23030                    CASE 1
23040                        X=P2*Scale
23050                    CASE 2
23060                        X=P3*Scale
23070                    END SELECT
23080                    Xm=MIN(MAX(X,Wndw(1)),Wndw(2))
23090                    Ym=MIN(MAX(Y,Wndw(3)),Wndw(4))
23100                    MOVE Xm,Ym
23110                    SYMBOL Symbol(*),FILL,EDGE
23120                NEXT I
23130            SUBEND
23140 Tcs8:      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
23150 Tcs8init:  SUB Tcs8init(@Tcs8)
23160                ! Description:
23170                !         This subprogram is used to initialize this computer's internal RS232 serial interface.
23180                !     The subprogram also opens the TCS8 path on the Hewlett Packard series 9000 model 3XX computer
```

```
23190                   !       for command and data transfer.  The I/O path is given the name "@Tcs8".  Data transferred
23200                   !       from the HP to the TCS8 will use the "OUTPUT @Tcs8" statement.  Data transferred to the HP
23210                   !       from TCS8 will use the "ENTER @Tcs8" statement.
23220                   !           The I/O path has a select code of 9 and is initialized to perform unformatted byte
23230                   !       transfers without any end of line designations.
23240                   ASSIGN @Tcs8 TO 9;BYTE,FORMAT OFF,EOL ""
23250                   CONTROL 9,0;1                ! Reset interface.
23260                   CONTROL 9,3;9600             ! Select a baud rate of 9600.
23270                   CONTROL 9,4;31               ! Select even parity, enable parity, 2 stop bits, 8 bits per character.
23280                   CONTROL 9,12;IVAL("EF",16)   ! Enable Carrier Detect.  Disable Data Set Ready.  Disable Clear To Send.
23290                   CONTROL 9,13;9600            ! Default baud rate of 9600.
23300                   CONTROL 9,14;31              ! Default character format: Even parity enabled, 2 stop, 8 bits/ char.
23310               SUBEND
23320 Tcs8set:      SUB Tcs8set(Command$,@Tcs8)
23330                   !   Description:
23340                   !       This subprogram allows the user to view and then set the various initialization parameters
23350                   !       of each channel of the TCS8.  These parameters are the current position, counts per inch,
23360                   !       counts per revolution, motor velocity, motor acceleration, plus and minus limit switches,
23370                   !       home switch, and motor stall indication.  All of these parameters can be viewed and set except
23380                   !       the limit and home switches and the stall indication.  They can only be viewed.
23390                   !   Variables:
23400                   !       Command$    A TCS8 command string which indicates which parameter we want to view & set.
23410                   !       View(*)     Array of old TCS8 parameters viewed (received from TCS8).  One for each channel.
23420                   !       Set(*)      Array of new TCS8 parameters to be set (sent to TCS8).  One for each channel.
23430                   !       Name$(*)    String array of TCS8 parameter names.
23440                   !       Image$(*)   String array of image formats.
23450                   !       Units$(*)   String array of units.
23460                   !       Channel     Indicates the TCS8 channel number.  Used to index the above arrays.
23470                   OPTION BASE 1
23480                   DIM View(8,1),Set(8,2),Name$(8,1)[10],Image$(8,1)[10],Units$(8,1)[10]
23490                   OUTPUT @Tcs8 USING "K,/";"V"&Command$&"0"     ! Tell the TCS8 we want to View a parameter.
23500                   ENTER @Tcs8 USING "8(K)";View(*)              ! Enter the parameter specified by Command$.
23510                   ! Initialize the Name$,Image$,Units$ and Set arrays.
23520                   READ Name$(*)
23530                   MAT Image$= ("6D.3D")
23540                   DATA X1,X2,Y1,Y2,Z1,Z2,A1,A2
23550                   FOR Channel=1 TO 8
23560                       Set(Channel,1)=Channel
23570                       SELECT Command$
23580                       CASE "P"     !  Command$="P" indicates we want to view the encoder Positions in inches.
23590                           Name$(Channel,1)=Name$(Channel,1)&" (pos)"
23600                           Units$(Channel,1)="in"
23610                       CASE "U"     !  Command$="U" indicates we want to view the Units in counts per inch.
23620                           Name$(Channel,1)=Name$(Channel,1)&" (cpi)"
23630                           Units$(Channel,1)="cnt"
23640                       CASE "R"     !  Command$="R" indicates we want to view the number counts per Revolution.
23650                           Name$(Channel,1)=Name$(Channel,1)&" (cpr)"
23660                           Units$(Channel,1)="cnt"
23670                       CASE "V"     !  Command$="V" indicates we want to view the Velocity in revolution per second.
23680                           Name$(Channel,1)=Name$(Channel,1)&" (vel)"
23690                           Units$(Channel,1)="rev"
23700                       CASE "A"     !  Command$="A" indicates we want to view the Acceleration in revolution per second^2.
23710                           Name$(Channel,1)=Name$(Channel,1)&" (acc)"
23720                           Units$(Channel,1)="rev"
23730                       CASE "+"     !  Command$="+" indicates we want to view the current + direction limit switches.
23740                           Name$(Channel,1)=Name$(Channel,1)&" (+LS)"
23750                           Units$(Channel,1)="   "
23760                       CASE "-"     !  Command$="-" indicates we want to view the current - direction limit switches.
23770                           Name$(Channel,1)=Name$(Channel,1)&" (-LS)"
23780                           Units$(Channel,1)="   "
23790                       CASE "S"     !  Command$="S" indicates we want to view the current motor Stall indication status.
23800                           Name$(Channel,1)=Name$(Channel,1)&" (STALL)"
23810                           Units$(Channel,1)="   "
23820                       CASE "H"     !  Command$="H" indicates we want to view the current Home limit switches.
23830                           Name$(Channel,1)=Name$(Channel,1)&" (HS)"
23840                           Units$(Channel,1)="   "
23850                       END SELECT
23860                   NEXT Channel
23870                   !  The "Change" subprogram allows the user to see and then change the values of the viewed parameters
23880                   CALL Change("VALUES",View(*),Name$(*),Image$(*),Units$(*))
23890                   !  The "Set" parameters command is now sent to the TCS8
23900                   SELECT Command$
23910                   CASE "P","U","R","V","A"
23920                       MAT Set(*,2)= View(*,1)
23930                       OUTPUT @Tcs8 USING 23940;"S"&Command$,Set(*)
23940                       IMAGE K,8(D,":",M6D.4D,","),/
23950                   END SELECT
23960               SUBEND
23970 Tcs8read:     SUB Tcs8read(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Tcs2tun1(*),Tcs2tun2(*),Tun2mod(*))
23980                   !   Description:
```

```
23990               !      This subprogram reads the current TCS8 positions.  The 8 positions are read in TCS
24000               !      coordinates with the units being in inches.  Four of the eight positions (X1,Y1,Z1,A1) which
24010               !      are the transmitting side traverse positions are entered into the Tcs1 array.  The other four
24020               !      positions (X2,Y2,Z2,A2) which are the receiving tide traverse positions are entered into the
24030               !      Tcs2 array.  The Tcs1 & Tcs2 arrays are converted from TCS to TUNNEL to MODEL coordinates.
24040               !      The current updated positions in the three coordinate systems are printed on the top of the
24050               !      CRT.  They are also returned to the main program.
24060               !  Variables:
24070               !      Tcs1(*)      TCS8 transmitting side traverse positions (X1,Y1,Z1,A1) in TCS coordinates.
24080               !      Tcs2(*)      TCS8  receiving   side traverse positions (X2,Y2,Z2,A2) in TCS coordinates.
24090               !      Tun(*)       Traverse positions (X,Y,Z) in TUNNEL coordinates.
24100               !      Mod(*)       Traverse positions (X,Y,Z) in MODEL  coordinates.
24110               !      Tcs2tun1(*)  Coordinate system transformation matrix for converting Tcs1(*) to Tun(*).
24120               !      Tcs2tun2(*)  Coordinate system transformation matrix for converting Tcs2(*) to Tun(*).
24130               !      Tun2mod(*)   Coordinate system transformation matrix for converting Tun(*)  to Mod(*).
24140               OUTPUT @Tcs8 USING "K,/";"VP0"
24150               ENTER @Tcs8 USING "8(K)";Tcs1(1),Tcs2(1),Tcs1(2),Tcs2(2),Tcs1(3),Tcs2(3),Tcs1(4),Tcs2(4)
24160               MAT Tun= Tcs2tun1*Tcs1
24170               REDIM Tun(1:3),Mod(1:3)
24180               MAT Mod= Tun2mod*Tun
24190               REDIM Tun(1:4),Mod(1:4)
24200               Mod(4)=0
24210               Tun(4)=0
24220               CALL Tcs8print(Mod(*),Tun(*),Tcs1(*),Tcs2(*))
24230             SUBEND
24240 Tcs8print:  SUB Tcs8print(Mod(*),Tun(*),Tcs1(*),Tcs2(*))
24250               !  Description:
24260               !      This subprogram prints the current updated TCS8 positions at the top of the CRT.  The
24270               !      positions are printed in TCS , TUNNEL , and MODEL coordinates.
24280               !  Variables:
24290               !      Tcs1(*)      TCS8 transmitting side traverse positions (X1,Y1,Z1,A1) in TCS coordinates.
24300               !      Tcs2(*)      TCS8  receiving   side traverse positions (X2,Y2,Z2,A2) in TCS coordinates.
24310               !      Tun(*)       Traverse positions (X,Y,Z) in TUNNEL coordinates.
24320               !      Mod(*)       Traverse positions (X,Y,Z) in MODEL  coordinates.
24330               PRINT CHR$(128);
24340               PRINT TABXY(50,1);"                                       "
24350               PRINT TABXY(50,2);"       MOD       TUN       TCS1      TCS2  "
24360               PRINT TABXY(50,3);"                                       "
24370               PRINT TABXY(50,4);
24380               PRINT USING "#,K,4(M3D.4D),X";" X:",Mod(1),Tun(1),Tcs1(1),Tcs2(1)
24390               PRINT TABXY(50,5);
24400               PRINT USING "#,K,4(M3D.4D),X";" Y:",Mod(2),Tun(2),Tcs1(2),Tcs2(2)
24410               PRINT TABXY(50,6);
24420               PRINT USING "#,K,4(M3D.4D),X";" Z:",Mod(3),Tun(3),Tcs1(3),Tcs2(3)
24430               PRINT TABXY(50,7);
24440               PRINT USING "#,K,4(M3D.4D),X";" A:",Mod(4),Tun(4),Tcs1(4),Tcs2(4)
24450               PRINT TABXY(50,8);"                                       "
24460             SUBEND
24470 Tcs8move:   SUB Tcs8move(@Tcs8,Mod(*),Tun(*),Tcs1(*),Tcs2(*),Mod2tun(*),Tun2tcs1(*),Tun2tcs2(*),Side$,Coor$,Mode$,
                          K,Movement)
24480               !  Description:
24490               !      This subprogram allows for the movement of the probe volume and collecting optics in one of
24500               !      three coordinate systems.  The three coordinate systems implemented are the TSC, TUNNEL and
24510               !      MODEL coordinate systems.  Two movements modes are available.  The first movement mode makes
24520               !      moves relative to the current position.  The second movement mode makes moves to an absolute
24530               !      fixed position.  Both the transmitting side and receiving side traverses can be moved in tandem
24540               !      or separately.
24550               !  Variables:
24560               !      Tcs1(*)      TCS8 transmitting side traverse positions (X1,Y1,Z1,A1) in TCS coordinates.
24570               !      Tcs2(*)      TCS8  receiving   side traverse positions (X2,Y2,Z2,A2) in TCS coordinates.
24580               !      Tun(*)       Traverse positions (X,Y,Z) in TUNNEL coordinates.
24590               !      Mod(*)       Traverse positions (X,Y,Z) in MODEL  coordinates.
24600               !      Mod2tun(*)   Coordinate system transformation matrix for converting Tcs1(*) to Tun(*).
24610               !      Tun2tcs1(*)  Coordinate system transformation matrix for converting Tcs2(*) to Tun(*).
24620               !      Tun2tcs2(*)  Coordinate system transformation matrix for converting Tcs2(*) to Tun(*).
24630               !      Side$        Indicates which sides are to be moved:
24640               !                       Tx      : Transmitting side only.
24650               !                       Rx      : Receiving   side only.
24660               !                       Tx & Rx : Both sides together.
24670               !      Coor$        Indicates which coordinate system the movement is to be made:
24680               !                       MODEL   : MODEL  coordinates.
24690               !                       TUNNEL  : TUNNEL coordinates.
24700               !                       TCS     : TCS    coordinates.
24710               !      Mode$        Indicates which movement mode is to be completed:
24720               !                       RELATIVE: Movements are relative to current positions.
24730               !                       ABSOLUTE: Movements are to absolute positions.
24740               !      K            Indicates which axis of the four axes is to be moved.
24750               !      Movement     Indicates the desired movement for the selected axis.
24760               !      I(*)         Array of viewed TCS8 "Initialized" parameters.
24770               !      C(*)         Array of viewed TCS8 "Currents On" parameters.
```

```
24780                       OPTION BASE 1
24790                       DIM L$[100]
24800                       REAL Move(8,2),I(8),C(8)
24810                       ! If all of the channels have not yet been initialized, then do so now.
24820                       OUTPUT @Tcs8 USING "K,/";"VIO"
24830                       ENTER @Tcs8 USING "8(K)";I(*)
24840                       IF SUM(I)<>8 THEN OUTPUT @Tcs8 USING "K,/";"SIO"
24850                       ! If all of the channels do not have their currents turned on, then do so now.
24860                       OUTPUT @Tcs8 USING "K,/";"VCO"
24870                       ENTER @Tcs8 USING "8(K)";C(*)
24880                       IF SUM(C)<>8 THEN OUTPUT @Tcs8 USING "K,/";"SCO:1,"
24890                       ! If the movement mode is to be RELATIVE, then clear all of the previously read positions.
24900                       IF Mode$="RELATIVE" THEN
24910                           MAT Mod= (0)
24920                           MAT Tun= (0)
24930                           MAT Tcs1= (0)
24940                           MAT Tcs2= (0)
24950                       END IF
24960                       ! Set the new Tcs1(*) and Tcs2(*) position arrays.
24970                       SELECT Coor$
24980                       CASE "MODEL"
24990                           Mod(K)=Movement
25000                           REDIM Tun(1:3),Mod(1:3)
25010                           MAT Tun= Mod2tun*Mod
25020                           REDIM Tun(1:4),Mod(1:4)
25030                           IF POS(Side$,"Tx") THEN MAT Tcs1= Tun2tcs1*Tun
25040                           IF POS(Side$,"Rx") THEN MAT Tcs2= Tun2tcs2*Tun
25050                       CASE "TUNNEL"
25060                           Tun(K)=Movement
25070                           IF POS(Side$,"Tx") THEN MAT Tcs1= Tun2tcs1*Tun
25080                           IF POS(Side$,"Rx") THEN MAT Tcs2= Tun2tcs2*Tun
25090                       CASE "TCS"
25100                           IF POS(Side$,"Tx") THEN Tcs1(K)=Movement
25110                           IF POS(Side$,"Rx") THEN Tcs2(K)=Movement
25120                       END SELECT
25130                       ! File the move array.
25140                       FOR Channel=1 TO 8
25150                           Move(Channel,1)=Channel
25160                       NEXT Channel
25170                       Move(1,2)=Tcs1(1)
25180                       Move(2,2)=Tcs2(1)
25190                       Move(3,2)=Tcs1(2)
25200                       Move(4,2)=Tcs2(2)
25210                       Move(5,2)=Tcs1(3)
25220                       Move(6,2)=Tcs2(3)
25230                       Move(7,2)=Tcs1(4)
25240                       Move(8,2)=Tcs2(4)
25250                       ! Initiate the start of the move.
25260                       IF Mode$="ABSOLUTE" THEN OUTPUT @Tcs8 USING 25280;"MA",Move(*)
25270                       IF Mode$="RELATIVE" THEN OUTPUT @Tcs8 USING 25280;"MR",Move(*)
25280                       IMAGE K,8(D,":",S2D.5D,","),/
25290                       ! The TCS8 will return the new updated positions only after the move is complete.
25300                       ENTER @Tcs8 USING "8(K)";Tcs1(1),Tcs2(1),Tcs1(2),Tcs2(2),Tcs1(3),Tcs2(3),Tcs1(4),Tcs2(4)
25310                       ! Turn off the motor drive currents.
25320                       OUTPUT @Tcs8 USING "K,/";"SCO:0,"
25330                   SUBEND
25340 Ctm:              !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
25350 Refract:          SUB Refract(REAL Index(*),Theta1(*),Theta3(*))
25360                       !  Description:
25370                       !      This subprogram uses the laser beam angles outside the tunnel to compute the angles inside
25380                       !      the water tunnel.  This requires the knowledge of the indexes of refraction for the various
25390                       !      media that the beams go through.  The Mediums are air, glass, and water.
25400                       !  Variables:
25410                       !      Index(*)    Array of indexes of refraction.
25420                       !                      Index(1) : Index of refraction for Air.
25430                       !                      Index(2) : Index of refraction for Glass.
25440                       !                      Index(3) : Index of refraction for Water.
25450                       !      Theta1(*)  Laser beam angles outside the water tunnel.
25460                       !      Theta3(*)  Laser beam angles inside  the water tunnel.
25470                       OPTION BASE 1
25480                       ! Correct Theta for angles in water.
25490                       MAT Theta3= Theta1
25500                       IF Index(1)<>Index(3) THEN
25510                           Theta3(2,1)=ASN(Index(1)/Index(3)*SIN(Theta1(2,1)))
25520                           Theta3(2,2)=ASN(Index(1)/Index(3)*SIN(Theta1(2,2)))+90
25530                       END IF
25540                   SUBEND
25550 Ctm_ldv:          SUB Ctm_ldv(Theta(*),Tun2ldv(*),Ldv2tun(*))
25560                       !  Description:
25570                       !      This subprogram computes directly the TUNNEL to LASER coordinate system transformation
```

```
25580                  !      matrix "Tun2ldv(*)".  However, the desired coordinate system transformation matrix "Ldv2tun" is
25590                  !      required.  It is the matrix inverse of "Tun2ldv".
25600                  !  Variables:
25610                  !      Theta(*)     Laser beam angles inside the water tunnel.
25620                  !      Tun2ldv(*)   Coordinate system transformation matrix for converting from TUNNEL to LASER.
25630                  !      Ldv2tun(*)   Coordinate system transformation matrix for converting from LASER to TUNNEL.
25640                  OPTION BASE 1
25650                  ! Tun2ldv converts TUNNEL coordinates to LASER coordinates.
25660                  Tun2ldv(1,1)=COS(Theta(1,1))
25670                  Tun2ldv(1,2)=COS(Theta(1,2))
25680                  Tun2ldv(1,3)=COS(Theta(1,3))
25690                  Tun2ldv(2,1)=COS(Theta(2,1))
25700                  Tun2ldv(2,2)=COS(Theta(2,2))
25710                  Tun2ldv(2,3)=COS(Theta(2,3))
25720                  Tun2ldv(3,1)=COS(Theta(3,1))
25730                  Tun2ldv(3,2)=COS(Theta(3,2))
25740                  Tun2ldv(3,3)=COS(Theta(3,3))
25750                  ! Ldv2tun converts LASER coordinates to TUNNEL coordinates.
25760                  MAT Ldv2tun= INV(Tun2ldv)
25770               SUBEND
25780 Ctm_mod:      SUB Ctm_mod(Alpha(*),Mod2tun(*),Tun2mod(*))
25790                  ! Description:
25800                  !      This subprogram computes directly the MODEL to TUNNEL coordinate system transformation
25810                  !      matrix "Mod2tun(*)".  However, the desired coordinate system transformation matrix "Tun2mod" is
25820                  !      required.  It is the matrix inverse of "Mod2tun".
25830                  !  Variables:
25840                  !      Alpha(*)     Angles of attack, yaw, and roll.
25850                  !      T1(*)        Partial coordinate system transformation matrix for converting from MODEL to
25860                  !                   TUNNEL coordinates.  Takes into account a model at angle of attack.
25870                  !      T2(*)        Partial coordinate system transformation matrix for converting from MODEL to
25880                  !                   TUNNEL coordinates.  Takes into account a model at angle of yaw.
25890                  !      T3(*)        Partial coordinate system transformation matrix for converting from MODEL to
25900                  !                   TUNNEL coordinates.  Takes into account a model at angle of roll.
25910                  !      Mod2tun(*)   Coordinate system transformation matrix for converting from MODEL to TUNNEL.
25920                  !      Tun2mod(*)   Coordinate system transformation matrix for converting from TUNNEL to MODEL.
25930                  OPTION BASE 1
25940                  REAL T1(3,3),T2(3,3),T3(3,3),Temp(3,3)
25950                  ! Define 1st coordinate transformation matrix for Mod2tun.
25960                  ! Rotation in the x-y plane about the z-axis.
25970                  ! Used when model is at an angle of attack.
25980                  T1(1,1)=COS(Alpha(1))
25990                  T1(1,2)=SIN(Alpha(1))
26000                  T1(1,3)=0
26010                  T1(2,1)=-SIN(Alpha(1))
26020                  T1(2,2)=COS(Alpha(1))
26030                  T1(2,3)=0
26040                  T1(3,1)=0
26050                  T1(3,2)=0
26060                  T1(3,3)=1
26070                  ! Define 2nd coordinate transformation matrix for Mod2tun.
26080                  ! Rotation in the x-z plane about the y-axis.
26090                  ! Used when model is at an angle of yaw.
26100                  T2(1,1)=COS(-Alpha(2))
26110                  T2(1,2)=0
26120                  T2(1,3)=-SIN(-Alpha(2))
26130                  T2(2,1)=0
26140                  T2(2,2)=1
26150                  T2(2,3)=0
26160                  T2(3,1)=SIN(-Alpha(2))
26170                  T2(3,2)=0
26180                  T2(3,3)=COS(-Alpha(2))
26190                  ! Define 3rd coordinate transformation matrix for Mod2tun.
26200                  ! Rotation in the y-z plane about the x-axis.
26210                  ! Used when model is at an angle of roll.
26220                  T3(1,1)=1
26230                  T3(1,2)=0
26240                  T3(1,3)=0
26250                  T3(2,1)=0
26260                  T3(2,2)=COS(-Alpha(3))
26270                  T3(2,3)=SIN(-Alpha(3))
26280                  T3(3,1)=0
26290                  T3(3,2)=-SIN(-Alpha(3))
26300                  T3(3,3)=COS(-Alpha(3))
26310                  ! Mod2tun converts MODEL coordinates to TUNNEL coordinates.
26320                  MAT Temp= T2*T1
26330                  MAT Mod2tun= T3*Temp
26340                  ! Tun2mod converts TUNNEL coordinates to MODEL coordinates.
26350                  MAT Tun2mod= INV(Mod2tun)
26360               SUBEND
26370 Ctm_tcs:      SUB Ctm_tcs(Tcs2tun1(*),Tcs2tun2(*),Tun2tcs1(*),Tun2tcs2(*),Fs,Fr,Bs,Br,Index(*),Ts,Tr,Ta)
```

```
26380              ! Description:
26390              !     This subprogram computes the TUNNEL to TCS coordinate system transformation matrices
26400              !     "Tun2tcs1(*)" and "Tun2tcs2(*)".  The coordinate system transformation matrices "Tcs2tun1" and
26410              !     "Tcs2tun1" are the matrix inverses of "Tun2tcs1(*)" and "Tun2tcs2(*)" respectively.
26420              ! Variables:
26430              !     Tcs2tun1(*)       Sending side coordinate transformation matrix converting Tcs(*) to Tun(*).
26440              !     Tun2tcs1(*)       Sending side coordinate transformation matrix converting Tun(*) to Tcs(*).
26450              !     Tcs2tun2(*)       Receiving side coordinate transformation matrix converting Tcs(*) to Tun(*).
26460              !     Tun2tcs2(*)       Receiving side coordinate transformation matrix converting Tun(*) to Tcs(*).
26470              !     Fs                Focal length for sending side onaxis and offaxis lenses.
26480              !     Fr                Focal length for receiving side offaxis lens.
26490              !     Bs                Beam spacings for sending side onaxis and offaxis beam pairs.
26500              !     Br                Beam spacing for receiving side offaxis.
26510              !     Index(*)          Array of indexes of refraction for air, glass, and water.
26520              !     Ts                Angle of offaxis sending side beam pair.
26530              !     Tr                Angle of offaxis receiving side beam pair.
26540              !     Ta                Sending side offaxis auxiliary traverse angle.  Returned to main program.
26550              !     Xs_on,Ys_on       Starting coordinates of onaxis  sending side lens.
26560              !     Xs_offs,Ys_offs   Starting coordinates of offaxis sending side lens.
26570              !     Xs_offr,Ys_offr   Starting coordinates of offaxis receiving side lens.
26580              !     Xc,Yc             The common point in air of two beam path equations.
26590              !     Ba,Bb             The Y intercepts of two beam path equations.
26600              !     Theta(*)          Array of angles in which each beam contacts the window.
26610              !     X(*)              Array of X coordinates for the points in which each beam contacts the window.
26620              !     Yposition         The Y coordinate of the point where all beams cross in the water.
26630              !     Y1,X2,Y2,X3,Y3    Temporary variables to hold the results of the first call to Findstart.
26640              !     Thickness         The thickness of the window.
26650              !     Beam              Subscript used while determining the X(*) array.
26660              OPTION BASE 1
26670              REAL Xs_on,Ys_on,Xs_offs,Ys_offs,Xs_offr,Ys_offr,Xc,Yc,Ba,Bb,Theta(6),X(6)
26680              REAL Yposition,Y1,X2,Y2,X3,Y3,Thickness
26690              INTEGER Beam
26700              Thickness=1.25
26710              Yposition=0
26720              GOSUB Findstart
26730              Y1=Ys_on
26740              X2=Xs_offs
26750              Y2=Ys_offs
26760              X3=Xs_offr
26770              Y3=Ys_offr
26780              Yposition=1
26790              GOSUB Findstart
26800              MAT Tun2tcs1= IDN
26810              MAT Tun2tcs2= IDN
26820              Tun2tcs1(2,2)=-Ys_on+Y1
26830              Tun2tcs1(4,2)=-SQRT((Xs_offs-X2)^2+(Ys_offs-(Ys_on-Y1+Y2))^2)
26840              Tun2tcs1(4,4)=0
26850              Ta=ATN((Xs_offs-X2)/(Ys_offs-(Ys_on-Y1+Y2)))
26860              Tun2tcs2(1,2)=Xs_offr-X3
26870              Tun2tcs2(2,2)=-Ys_offr+Y3
26880              Tun2tcs2(4,4)=0
26890              MAT Tcs2tun1= INV(Tun2tcs1)
26900              MAT Tcs2tun2= INV(Tun2tcs2)
26910              Tcs2tun1(4,2)=0
26920              Tcs2tun2(4,2)=0
26930              SUBEXIT
26940 Findstart:   !     This subroutine finds the starting coordinates for the onaxis (Xs_on,Ys_on) and offaxis (Xs_offs,
26950              ! Ys_offs) sending side lenses and the offaxis (Xs_offr,Ys_offr) receiving side lens given the point
26960              ! at which all beams cross in the tunnel.  The crossing point is given to be (0,Yposition).  The
26970              ! method in which the starting coordinates are found involves solving simultaneously the equations for
26980              ! the path of each beam pair in air yielding the common point (Xc,Yc).  Given the focal length of the
26990              ! lens, the starting coordinate can be calculated.  The equation for each beam path in air is obtained
27000              ! by determining the angle and the point a beam contacts the window.
27010              !
27020              ! These six equations find the six angles.
27030              Theta(1)=ATN(Bs/(2*Fs))
27040              Theta(2)=-ATN(Bs/(2*Fs))
27050              Theta(3)=Ts+ATN(Bs/(2*Fs))
27060              Theta(4)=Ts-ATN(Bs/(2*Fs))
27070              Theta(5)=Tr+ATN(Br/(2*Fr))
27080              Theta(6)=Tr-ATN(Br/(2*Fr))
27090              ! This equation finds the X coordinate of the six points.  The Y coordinate is equal to -Thickness.
27100              FOR Beam=1 TO 6
27110                X(Beam)=-Yposition*TAN(ASN(Index(1)/Index(3)*SIN(Theta(Beam))))-
                           Thickness*TAN(ASN(Index(1)/Index(2)*SIN(Theta(Beam))))
27120              NEXT Beam
27130              ! Determine the Y intercepts for the onaxis beam paths.
27140              Ba=-Thickness-X(1)/TAN(Theta(1))
27150              Bb=-Thickness-X(2)/TAN(Theta(2))
27160              ! Solve for the common point.
```

```
27170                   Xc=(Bb-Ba)/(1/TAN(Theta(1))-1/TAN(Theta(2)))
27180                   Yc=Xc/TAN(Theta(2))+Bb
27190                   ! Calculate the onaxis lens starting coordinate using the focal length and the onaxis angle (=0 deg).
27200                   Xs_on=Xc-Fs*SIN(0)
27210                   Ys_on=Yc-Fs*COS(0)
27220                   ! Determine the Y intercepts for the offaxis sending side beam paths.
27230                   Ba=-Thickness-X(3)/TAN(Theta(3))
27240                   Bb=-Thickness-X(4)/TAN(Theta(4))
27250                   ! Solve for the common point.
27260                   Xc=(Bb-Ba)/(1/TAN(Theta(3))-1/TAN(Theta(4)))
27270                   Yc=Xc/TAN(Theta(4))+Bb
27280                   ! With the focal length and the offaxis angle calculate the starting coordinate
27290                   ! of the offaxis sending side lens.
27300                   Xs_offs=Xc-Fs*SIN(Ts)
27310                   Ys_offs=Yc-Fs*COS(Ts)
27320                   ! Determine the Y intercepts for the offaxis receiving side beam paths.
27330                   Ba=-Thickness-X(5)/TAN(Theta(5))
27340                   Bb=-Thickness-X(6)/TAN(Theta(6))
27350                   ! Solve for the common point.
27360                   Xc=(Bb-Ba)/(1/TAN(Theta(5))-1/TAN(Theta(6)))
27370                   Yc=Xc/TAN(Theta(6))+Bb
27380                   ! With the focal length and the offaxis angle calculate the starting coordinate
27390                   ! of the offaxis receiving side lens.
27400                   Xs_offr=Xc-Fr*SIN(Tr)
27410                   Ys_offr=Yc-Fr*COS(Tr)
27420                   RETURN
27430               SUBEND
27440 Graph:        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
27450 Crt_init:     SUB Crt_init
27460                   !  Description:
27470                   !      This subprogram initializes the CRT as the plotting device and clears both the alpha
27480                   !      numerics and graphics part of the CRT.
27490                   PLOTTER IS CRT,"INTERNAL"     ! Select the CRT as the plotting device.
27500                   AREA PEN 0                    ! Select black for area fills.
27510                   PEN 1                         ! Select white for line drawing and labeling.
27520                   PRINTER IS CRT                ! Select the CRT as the printing device.
27530                   PRINTALL IS CRT               ! Send ERROR and DISP message to CRT.
27540                   KEY LABELS OFF                ! Hide the special function key labels for f1..f8.
27550               SUBEND
27560 Read_symbols: SUB Read_symbols(Symbols(*))
27570                   !  Description:
27580                   !      This subprogram defines 5 geometric symbols to be used with the SYMBOL statement.  The
27590                   !      symbols provided are as follows:  Square,Octagon,Diamond, and Triangles (upwards & downwards
27600                   !      pointing triangles).  All of the symbols have a dot added to their center.
27610                   !  Variables:
27620                   !      Symbols(*)  Array of Symbol arrays.  Each symbol arrays contains a distinct geometric symbol.
27630                   !      Symbol(*)   Array of coordinates which when connected produce a distinct geometric symbol.
27640                   !      Dot(*)      Array of coordinates which produce a dot.  The dot symbol is added to all symbols.
27650                   !      Noc         The number of coordinates in a symbol.
27660                   !      S           Used to index the Symbols array.
27670                   OPTION BASE 1
27680                   REAL Symbol(20,3),Dot(2,3)
27690                   READ Dot(*)
27700                   FOR S=1 TO 5
27710                       READ Noc
27720                       REDIM Symbol(Noc,3)
27730                       READ Symbol(*)
27740                       MAT Symbols(S,1:Noc,*)= Symbol
27750                       MAT Symbols(S,Noc+1:Noc+2,*)= Dot
27760                       Symbols(S,0,1)=Noc+2
27770                   NEXT S
27780 Dot:              DATA     4.5, 7.5,-2,  4.5, 7.5,-1
27790 Square:           DATA 5,  0.5, 3.5,-2,  8.5, 3.5,-1,  8.5,11.5,-1,  0.5,11.5,-1,  0.5,3.5,-1
27800 Octagon:          DATA 9,  0.5, 5.5,-2,  2.5, 3.5,-1,  6.5, 3.5,-1,  8.5, 5.5,-1,  8.5,9.5,-1,  6.5,11.5,-1,
                                  2.5,11.5,-1,  0.5,9.5,-1,  0.5,5.5,-1
27810 Diamond:          DATA 5, -0.5, 7.5,-2,  4.5, 2.5,-1,  9.5, 7.5,-1,  4.5,12.5,-1, -0.5,7.5,-1
27820 Utriangle:        DATA 4,  0.5, 4.5,-2,  8.5, 4.5,-1,  4.5,13.5,-1,  0.5, 4.5,-1
27830 Dtriangle:        DATA 4,  0.5,10.5,-2,  8.5,10.5,-1,  4.5, 1.5,-1,  0.5,10.5,-1
27840               SUBEND
27850 Setup_graph:  SUB Setup_graph(Array(*),Image$(*),Paxis,Symbols(*))
27860                   !  Description:
27870                   !      This subprogram sets up nine empty plots on the CRT screen.  Four plots are profile plots
27880                   !      while the other five plots are histogram plots.  The profile and histogram plots provided are
27890                   !      as follows:      Graph#        Type              Description
27900                   !                         1       Histogram #1      U frequency data in MHz.
27910                   !                         2       Histogram #2      V frequency data in MHz.
27920                   !                         3       Histogram #3      W frequency data in MHz.
27930                   !                         4       Histogram #4      Analog Channel #1 data in volts.
27940                   !                         5       Histogram #5      Analog Channel #2 data in volts.
27950                   !                         6       Profile Plot #1   Velocity Averages versus Traverse Position.
```

```
27960          !                         7       Profile Plot #2     Velocity SDVs versus Traverse Position.
27970          !                         8       Profile Plot #3     Velocity Shear Stresses versus Traverse Position.
27980          !                         9       Profile Plot #4     Average voltages & SDVs versus Traverse Position.
27990          !  Variables:
28000          !      Array(*)      Array containing the plot positions and scales.
28010          !      Image$(*)     String array containing image formats for the axes labeling.
28020          !      Wndw(*)       Array containing the plot's scales.
28030          !      Vwprt(*)      Array containing the plot's CRT position.
28040          !      Xdiv(*)       Array containing the number of X divisions for the plot's X axis.
28050          !      Ydiv(*)       Array containing the number of Y divisions for the plot's Y axis.
28060          !      Xlabel$(*)    String array containing labels for the X axis.
28070          !      Ylabel$(*)    String array containing labels for the Y axis.
28080          !      Title$(*)     String array containing labels for the Plots.
28090          !      Ximage$(*)    String array containing image formats for the X axis labeling.
28100          !      Yimage$(*)    String array containing image formats for the Y axis labeling.
28110          !      Legend$(*)    String array containing labels for each symbol in a profile plot.
28120          !      Symbols(*)    Array of Symbol arrays.  Each symbol arrays contains a distinct geometric symbol.
28130          !      G             Used as an index to the above arrays.  Specifies one of nine plots.
28140          !      I             Used an an index to the Legend$ array.
28150          OPTION BASE 1
28160          COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                        Legend$(*)
28170          MAT Wndw= Array(61:69,*)
28180          MAT Vwprt= Array(71:79,*)
28190          MAT Xdiv(1:5)= Array(81:85,1)
28200          MAT Xdiv(6:9)= Array(81:84,3)
28210          MAT Ydiv(1:5)= Array(81:85,2)
28220          MAT Ydiv(6:9)= Array(81:84,4)
28230          MAT Ximage$= Image$(61:69,1)
28240          MAT Yimage$= Image$(61:69,3)
28250          FOR G=1 TO 9
28260              READ G,Xlabel$(G)
28270              FOR I=1 TO SIZE(Legend$,2)
28280                  READ Legend$(G,I)
28290              NEXT I
28300              SELECT G
28310              CASE 1 TO 5
28320                  Ylabel$(G)=""
28330              CASE 6 TO 9
28340                  Ylabel$(G)=CHR$(NUM("X")+Paxis-1)
28350              END SELECT
28360              CALL Set_up(G,Symbols(*))
28370          NEXT G
28380          SUBEXIT
28390          !    G, X axis Label                     ,      Symbol #1...5 labels
28400          DATA 1, ""                                ,    "",     "",      "","",""
28410          DATA 2, ""                                ,    "",     "",      "","",""
28420          DATA 3, ""                                ,    "",     "",      "","",""
28430          DATA 4, ""                                ,    "",     "",      "","",""
28440          DATA 5, ""                                ,    "",     "",      "","",""
28450          DATA 6, "Velocities / Uedge"              ,   "U",    "V",     "W","",""
28460          DATA 7, "RMS / Uedge"                     ,  "U1",   "V1",    "W1","",""
28470          DATA 8, "Shear Stress / Uedge^2"          ,"U1v1","V1w1",  "W1u1","",""
28480          DATA 9, "Fluorescence:   C,C1 (volts)"    ,   "C",     "",    "C1","",""
28490      SUBEND
28500 Set_up:  SUB Set_up(G,Symbols(*))
28510          !  Description:
28520          !      This subprogram clears and then redraws one of nine empty plots on the CRT screen.
28530          !  Variables:
28540          !      Wndw(*)       Array containing the plot's scales.
28550          !      Vwprt(*)      Array containing the plot's CRT position.
28560          !      Xdiv(*)       Array containing the number of X divisions for the plot's X axis.
28570          !      Ydiv(*)       Array containing the number of Y divisions for the plot's Y axis.
28580          !      Xlabel$(*)    String array containing labels for the X axis.
28590          !      Ylabel$(*)    String array containing labels for the Y axis.
28600          !      Title$(*)     String array containing labels for the Plots.
28610          !      Ximage$(*)    String array containing image formats for the X axis labeling.
28620          !      Yimage$(*)    String array containing image formats for the Y axis labeling.
28630          !      Legend$(*)    String array containing labels for each symbol in a profile plot.
28640          !      Symbols(*)    Array of Symbol arrays.  Each symbol arrays contains a distinct geometric symbol.
28650          !      G             Used as an index to the above arrays.  Specifies one of nine plots.
28660          OPTION BASE 1
28670          COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                        Legend$(*)
28680          DIM L$[80]
28690          ON ERROR CALL Error
28700          PLOTTER IS CRT,"INTERNAL"
28710          ! Define the pen numbers for the colors black and white.
28720          Black=-1
28730          White=1
```

```
28740                CSIZE 100*15/1023      ! Select a character labeling size of 15 pixels high.
28750                ! Define the values for the left,right,bottom,top ends of the horizontal and vertical scales.
28760                Xmin=Wndw(G,1)
28770                Xmax=Wndw(G,2)
28780                Ymin=Wndw(G,3)
28790                Ymax=Wndw(G,4)
28800                ! Define the values for the left,right,bottom,top pixel locations for the plot.
28810                Xpix1=Vwprt(G,1)
28820                Xpix2=Vwprt(G,2)
28830                Ypix1=Vwprt(G,3)
28840                Ypix2=Vwprt(G,4)
28850                ! Define the step size between grid lines, axis tick marks, and axis labels.
28860                Xstep=(Xmax-Xmin)/Xdiv(G)
28870                Ystep=(Ymax-Ymin)/Ydiv(G)
28880                ! Define the amount of scale X and Y which equals the size of one pixel (picture element).
28890                Xpixel=(Xmax-Xmin)/(Xpix2-Xpix1)
28900                Ypixel=(Ymax-Ymin)/(Ypix2-Ypix1)
28910                ! Clear the plots back ground & plot area and also draw the plots borders, grids, and axes.
28920                AREA PEN Black
28930                PEN White
28940                GOSUB Back_ground
28950                GOSUB Axes
28960               !GOSUB Grid
28970                GOSUB Plot_area
28980                ! Draw the X and Y axis labels.
28990                CLIP OFF
29000                GOSUB Ylabel
29010                GOSUB Xlabel
29020                ! Create a legend to define which symbol is used with which data.
29030                CALL Legend(G,Symbols(*))
29040                OFF ERROR
29050                SUBEXIT
29060 Back_ground:   ! This subroutine clears the plot's background.
29070                VIEWPORT (Xpix1-75)/10.23,(Xpix2+25)/10.23,(Ypix1-33)/10.23,(Ypix2+6)/10.23
29080                WINDOW -1.E+9,1.E+9,-1.E+9,1.E+9
29090                MOVE 0,0
29100                WINDOW 0,1,0,1
29110                MOVE 0,0
29120                RECTANGLE 1,1,FILL
29130                RETURN
29140 Axes:          ! This subroutine draws the plot's X and Y axes.
29150                VIEWPORT (Xpix1-1)/10.23,(Xpix2+1)/10.23,(Ypix1-6)/10.23,(Ypix1-1)/10.23
29160                WINDOW Xmin,Xmax,1,0
29170                AXES Xstep,2,Xmin,0,1,1,1
29180                VIEWPORT (Xpix1-1)/10.23,(Xpix2+1)/10.23,(Ypix2+1)/10.23,(Ypix2+6)/10.23
29190                WINDOW Xmin,Xmax,0,1
29200                AXES Xstep,2,Xmin,0,1,1,1
29210                VIEWPORT (Xpix1-6)/10.23,(Xpix1-1)/10.23,(Ypix1-1)/10.23,(Ypix2+1)/10.23
29220                WINDOW 1,0,Ymin,Ymax
29230                AXES 2,Ystep,0,Ymin,1,1,1
29240                VIEWPORT (Xpix2+1)/10.23,(Xpix2+6)/10.23,(Ypix1-1)/10.23,(Ypix2+1)/10.23
29250                WINDOW 0,1,Ymin,Ymax
29260                AXES 2,Ystep,0,Ymin,1,1,1
29270                RETURN
29280 Grid:          ! This subroutine draws the plot's X and Y grid lines.
29290                VIEWPORT (Xpix1-1)/10.23,(Xpix2+1)/10.23,(Ypix1-1)/10.23,(Ypix2+1)/10.23
29300                WINDOW Xmin,Xmax,Ymin,Ymax
29310                LINE TYPE 4
29320                GRID Xstep,Ystep,Xmin,Ymin
29330                LINE TYPE 1
29340                RETURN
29350 Plot_area:     ! This subroutine selects part of the CRT plot area and give it scales for the X and Y axes.
29360                VIEWPORT Xpix1/10.23,Xpix2/10.23,Ypix1/10.23,Ypix2/10.23
29370                WINDOW Xmin,Xmax,Ymin,Ymax
29380                RETURN
29390 Xlabel:        ! This subroutine labels the X axis and also names the X axis.
29400                LORG 5
29410                FOR X=Xmin TO Xmax+Xstep/100 STEP Xstep
29420                    MOVE X,Ymin-12*Ypixel
29430                    OUTPUT L$ USING Ximage$(G);X
29440                    LABEL TRIM$(L$)
29450                NEXT X
29460                MOVE (Xmin+Xmax)/2,Ymin-25*Ypixel
29470                LABEL Xlabel$(G)
29480                RETURN
29490 Ylabel:        ! This subroutine labels the Y axis and also names the Y axis.
29500                LORG 8
29510                Len=0
29520                FOR Y=Ymin TO Ymax+Ystep/100 STEP Ystep
29530                    MOVE Xmin-5*Xpixel,Y
```

```
29540                       OUTPUT L$ USING Yimage$(G);Y
29550                       LABEL TRIM$(L$)
29560                       Len=MAX(Len,LEN(TRIM$(L$)))
29570                   NEXT Y
29580                   MOVE Xmin-(5+7*Len)*Xpixel,(Ymin+Ymax)/2
29590                   LABEL Ylabel$(G)
29600                   LORG 5
29610                   RETURN
29620               SUBEND
29630 Legend:       SUB Legend(G,Symbols(*))
29640                   !  Description:
29650                   !      This subprogram produces a legend within one of the nine plots on the CRT screen.
29660                   !  Variables:
29670                   !      Wndw(*)       Array containing the plot's scales.
29680                   !      Vwprt(*)      Array containing the plot's CRT position.
29690                   !      Xdiv(*)       Array containing the number of X divisions for the plot's X axis.
29700                   !      Ydiv(*)       Array containing the number of Y divisions for the plot's Y axis.
29710                   !      Xlabel$(*)    String array containing labels for the X axis.
29720                   !      Ylabel$(*)    String array containing labels for the Y axis.
29730                   !      Title$(*)     String array containing labels for the Plots.
29740                   !      Ximage$(*)    String array containing image formats for the X axis labeling.
29750                   !      Yimage$(*)    String array containing image formats for the Y axis labeling.
29760                   !      Legend$(*)    String array containing labels for each symbol in a profile plot.
29770                   !      Symbols(*)    Array of Symbol arrays.  Each symbol arrays contains a distinct geometric symbol.
29780                   !      Symbol(*)     Array of coordinates which when connected produce a distinct geometric symbol.
29790                   !      G             Used as an index to the above arrays.  Specifies one of nine plots.
29800                   !      S             Used to index the Legend$ array.
29810                   !      Noc           The number of coordinates in a symbol.
29820                   !      Len           Length of a Legend$ array element.
29830                   OPTION BASE 1
29840                   COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                                 Legend$(*)
29850                   DIM Symbol(20,3)
29860                   VIEWPORT Vwprt(G,1)/10.23,Vwprt(G,2)/10.23,Vwprt(G,3)/10.23,Vwprt(G,4)/10.23
29870                   WINDOW Vwprt(G,1),Vwprt(G,2),Vwprt(G,3),Vwprt(G,4)
29880                   ! Define the pen numbers for the colors black and white.
29890                   Black=-1
29900                   White=1
29910                   ! Define the colors for symbol filling and edge drawing.
29920                   AREA PEN Black
29930                   PEN White
29940                   CSIZE 100*15/1023       ! Select a character labeling size of 15 pixels high.
29950                   LORG 2
29960                   ! Calculate the maximum length of all of the symbol labels.
29970                   Len=0
29980                   FOR S=1 TO SIZE(Legend$,2)
29990                       Len=MAX(LEN(Legend$(G,S)),Len)
30000                   NEXT S
30010                   ! For each symbol put up a sample symbol and its label.
30020                   FOR S=1 TO SIZE(Legend$,2)
30030                       IF LEN(Legend$(G,S))=0 THEN 30110
30040                       Noc=Symbols(S,0,1)
30050                       REDIM Symbol(Noc,3)
30060                       MAT Symbol= Symbols(S,1:Noc,*)
30070                       MOVE Vwprt(G,2)-7*Len-23,Vwprt(G,4)-15*S+5
30080                       SYMBOL Symbol(*),FILL,EDGE
30090                       MOVE Vwprt(G,2)-7*Len-10,Vwprt(G,4)-15*S+4
30100                       LABEL Legend$(G,S)
30110                   NEXT S
30120               SUBEND
30130 Histo:        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
30140 Rt_histo:     SUB Rt_histo(@Lvdas,Symbols(*),Repeat)
30150                   !  Description:
30160                   !      This subprogram plots real time histograms within five of the nine plots on the CRT screen.
30170                   !      The histogram data is acquired from the LVDAS over a specified acquisition time.
30180                   !  Variables Defined in Main Program:
30190                   !     Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),Legend$(*)
30200                   !  Local Variables:
30210                   !     Histo(*)  Array of bin numbers, old histogram bin heights, and new histogram bin heights.
30220                   !     Nbins     Number of bins in the Histo(*).
30230                   !     Bin       2^Bin is the bin width of individual histogram vertical bars.
30240                   !     Min       Minimum value for histogram.  Left  side of histogram scale.
30250                   !     Max       Maximum value for histogram.  right side of histogram scale.
30260                   !     F1        Upper 16bits of integerized Min.
30270                   !     F2        Lower 16bits of integerized Min.
30280                   !     A1        Upper 16bits of integerized histogram acquisition time.
30290                   !     A2        Lower 16bits of integerized histogram acquisition time.
30300                   !     Nnew      Number of samples in the most up to date histogram.
30310                   !     Nold      Number of samples in the previous histogram.
30320                   !     N(*)      Number of samples for each histogram of the five separate channels.
```

```
30330         !      Channel   Used to select the LVDAS channel that will be sampled for a histogram.
30340         !      Kw        Converts Hz to MHz or raw data to volts.
30350         !      Ww        Window width of each vertical histogram bar.
30360         !      Old       Histogram height of previous histogram at a particular bin.
30370         !      New       Histogram height of current  histogram at a particular bin.
30380         !      X1        Horizontal position of histogram rectangle.
30390         !      Y1        Vertical  position of histogram rectangle.
30400         !      X2        Horizontal width of histogram rectangle.
30410         !      Y2        Vertical  width of histogram rectangle.
30420         !      I         Used as an index to the Histo(*).  Specifies one of Nbins bins.
30430         !      G         Used as an index to the graphics arrays.  Specifies one of nine plots.
30440         OPTION BASE 1
30450         COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                          Legend$(*)
30460         INTEGER Histo(1000,3),Nbins,F1,F2,A1,A2
30470         REAL Nnew,Nold,N(5)
30480         ! Clear all of the histogram data within the LVDAS.
30490         OUTPUT @Lvdas USING "AA";"CA"
30500         ! Draw new plots for the five histograms.
30510         FOR Channel=1 TO 5
30520             CALL Set_up(Channel,Symbols(*))
30530         NEXT Channel
30540         ! Calculate the acquisition time.  0.1*10000000 will give an acquisition of 0.1 seconds.
30550         CALL Convert2words(.1*10000000,A1,A2)
30560         ! Enable the keyboard to terminate histogram plotting.
30570         ON KBD GOSUB Hdone
30580         REPEAT
30590             FOR Channel=1 TO 5
30600                 G=Channel
30610                 SELECT Channel
30620                 CASE 1,2,3                                 ! Channels 1,2,3 are for LDV frequency data.
30630                     Kw=1000000                             ! Converts Hz to MHz.
30640                     Min=Kw*Wndw(G,1)                       ! Minimum frequency for left  histogram scale.
30650                     Max=Kw*Wndw(G,2)                       ! Maximum frequency for right histogram scale.
30660                     Bin=INT(LGT((Max-Min)/100)/LGT(2))+1   ! 2^Bin is the window width of each vertical bars.
30670                     Ww=2^Bin                               ! Window width of each vertical histogram bar.
30680                     CALL Convert2words(Min,F1,F2)
30690                 CASE 4,5                                   ! Channels 4,5 are for analog voltage data.
30700                     Kw=32768/5                             ! Converts raw data to volts.
30710                     Min=Kw*Wndw(G,1)                       ! Minimum voltage for left  histogram scale.
30720                     Max=Kw*Wndw(G,2)                       ! Maximum voltage for right histogram scale.
30730                     Bin=INT(LGT((Max-Min)/100)/LGT(2))+1   ! 2^Bin is the window width of each vertical bars.
30740                     Ww=2^Bin                               ! Window width of each vertical histogram bar.
30750                     CALL Convert2words(Min,F1,F2)
30760                 CASE ELSE
30770                     GOTO 31100
30780                 END SELECT
30790 Hsend:          ! Tell the LVDAS to Take a Histogram.
30800                 OUTPUT @Lvdas USING "AA,6(W)";"TH",F1,F2,Bin,A1,A2,Channel
30810 Henter:         ! Enter number of bins in the histogram.
30820                 ENTER @Lvdas USING "#,W";Nbins
30830                 ! Redimension the Histo(*) and the enter the histogram data.
30840                 IF Nbins>0 THEN
30850                     REDIM Histo(Nbins,3)
30860                     ENTER @Lvdas USING "#,W";Histo(*)
30870                 END IF
30880                 ! Enter the number of samples for the previous and current histogram.
30890                 ENTER @Lvdas USING "#,W";Nnew,Nold
30900 Hplot:          ! Scale part of the CRT for the histogram plotting.
30910                 VIEWPORT Vwprt(G,1)/10.23,Vwprt(G,2)/10.23,Vwprt(G,3)/10.23,Vwprt(G,4)/10.23
30920                 WINDOW Kw*Wndw(G,1),Kw*Wndw(G,2),Wndw(G,3),Wndw(G,4)
30930                 Xpixel=Kw*(Wndw(Channel,2)-Wndw(Channel,1))/(Vwprt(Channel,2)-Vwprt(Channel,1))
30940                 N1=N(Channel)
30950                 N2=N(Channel)-Nold+Nnew
30960                 N(Channel)=N(Channel)-Nold+Nnew
30970                 FOR I=1 TO Nbins
30980                     Old=MIN(Histo(I,3),Wndw(Channel,4))
30990                     New=MIN(Histo(I,2),Wndw(Channel,4))
31000                     AREA PEN SGN(New-Old)          ! Positive pens will plot while negative histograms erase.
31010                     X1=Histo(I,1)*Ww+Min           ! Calculate histogram bar horizontal position.
31020                     X2=Ww                          ! Calculate histogram bar horizontal width.
31030                     Y1=Old                         ! Calculate histogram bar vertical position.
31040                     Y2=New-Old                     ! Calculate histogram bar vertical width.
31050                     IF X1<Kw*Wndw(G,1) THEN X1=Kw*Wndw(G,1)           ! If X1<Xmin then set X1=Xmin
31060                     IF X1>Kw*Wndw(G,2)-X2 THEN X1=Kw*Wndw(G,2)-X2     ! If X1>Xmax then set X1=Xmax
31070                     MOVE X1,Y1
31080                     RECTANGLE X2-Xpixel,Y2,FILL    ! Draw the rectangle representing one bar of the bar graph.
31090                 NEXT I
31100             NEXT Channel
31110         UNTIL KBD$<>"" OR NOT Repeat           ! Quit if any key on the keyboard has been pressed.
```

```
31120               SUBEXIT
31130 Hdone:        Done=1
31140               RETURN
31150           SUBEND
31160 Histo:    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
31170 Pt_histo:   SUB Pt_histo(Symbols(*),Run,File,Pos,INTEGER Nsam)
31180               ! Description:
31190               !     This subprogram plots post time histograms within five of the nine plots on the CRT screen.
31200               !     The histogram data is acquired from the LVDAS over a specified acquisition time.
31210               ! Variables Defined in Main Program:
31220               !    Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),Legend$(*)
31230               !    Ui(*),Vi(*),Wi(*),Ai(*),Bi(*)
31240               ! Local Variables:
31250               !    Histo(*)  Array of histogram bin heights indexed by bin number.
31260               !    Data(*)   Array of instantaneous U,V,W velocity or A,B voltage data.
31270               !    Nsam      Number of samples acquired.
31280               !    Xmin      Minimum value for histogram.  Left  side of histogram scale.
31290               !    Xmax      Maximum value for histogram.  right side of histogram scale.
31300               !    Xwin      Window width of each vertical histogram bar.
31310               !    K         Used as an index to the above arrays.
31320               !    L         Used as an index to the Histo(*).  Specifies one of 100 bins.
31330               !    Xpixel    Horizontal length of one picture on the CRT in scale units.
31340               !    Channel   Selects one of the 5 channels of Ui(*),Vi(*),Wi(*),Ai(*),Bi(*) data.
31350               !    G         Used as an index to the graphics arrays.  Specifies one of nine plots.
31360               OPTION BASE 1
31370               COM /Data/ INTEGER Raw(*),Valid(*),REAL Table(*),Ui(*),Vi(*),Wi(*),Ai(*),Bi(*),Ii(*),Ci(*)
31380               COM /Graph/ Wndw(*),Vwprt(*),Xdiv(*),Ydiv(*),Xlabel$(*),Ylabel$(*),Title$(*),Ximage$(*),Yimage$(*),
                           Legend$(*)
31390               INTEGER Histo(0:100)
31400               REAL Data(1000)
31410               REDIM Data(Nsam)
31420               FOR Channel=1 TO 5
31430                   ! Fill the data array with Ui(*),Vi(*),Wi(*),Ai(*), or Bi(*) depending on Channel.
31440                   G=Channel
31450                   IF Channel=1 THEN MAT Data= Ui
31460                   IF Channel=2 THEN MAT Data= Vi
31470                   IF Channel=3 THEN MAT Data= Wi
31480                   IF Channel=4 THEN MAT Data= Ai
31490                   IF Channel=5 THEN MAT Data= Bi
31500                   ! Draw a new empty histogram plot.
31510                   CALL Set_up(Channel,Symbols(*))
31520 Hsort:            Xmin=Wndw(Channel,1)
31530                   Xmax=Wndw(Channel,2)
31540                   Xwin=(Xmax-Xmin)/100
31550                   ! Sort the data into a histogram.
31560                   MAT Data= Data-(Xmin)
31570                   MAT Data= Data/((Xmax-Xmin)/100)
31580                   MAT Histo= (0)
31590                   FOR K=1 TO Nsam
31600                       L=MAX(MIN(Data(K),100),0)
31610                       Histo(L)=Histo(L)+1
31620                   NEXT K
31630 Hplot:            ! Scale part of the CRT for histogram plotting.
31640                   VIEWPORT Vwprt(G,1)/10.23,Vwprt(G,2)/10.23,Vwprt(G,3)/10.23,Vwprt(G,4)/10.23
31650                   WINDOW 0,100,Wndw(G,3),Wndw(G,4)
31660                   Xpixel=(100-0)/(Vwprt(Channel,2)-Vwprt(Channel,1))
31670                   ! Draw the histogram.
31680                   FOR K=0 TO 100
31690                       IF Histo(K) THEN
31700                           MOVE K-.5,0
31710                           AREA PEN SGN(1)    ! Positive pens will plot while negative histograms erase.
31720                           RECTANGLE 1-Xpixel,Histo(K),FILL
31730                       END IF
31740                   NEXT K
31750               NEXT Channel
31760               SUBEXIT
31770           SUBEND
DONE
```

# REPORT DOCUMENTATION PAGE

*Form Approved OMB No. 0704-0188*



1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
January 1992

3. REPORT TYPE AND DATES COVERED
Final Contractor Report

4. TITLE AND SUBTITLE
Measurement of Vortex Flow Fields

5. FUNDING NUMBERS
C NAS1-18667
WU 505-62-30-01

6. AUTHOR(S)
T. Kevin McDevitt, Todd A. Ambur, Gary M. Orngard, F. Kevin Owen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Complere, Inc.
P.O. Box 1697
Palo Alto, CA 94302

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-189543

11. SUPPLEMENTARY NOTES
Technical Monitor - Bobby L. Berrier, MS 280, Langley Research Center, Hampton, VA 23665
(804) 864-3001 FTS 928-3001

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A three-dimensional laser fluorescence anemometer has been designed, built, and demonstrated for use in the Langley 16-by 24-inch Water Tunnel. Innovative optical design flexibility combined with compact and portable data acquisition and control systems have been incorporated into the instrument. This will allow its use by NASA in other test facilities. A versatile fiber optic system facilitates normal and off-axis laser beam alignment, removes mirror losses and improves laser safety. This added optical flexibility will also enable simple adaptation for use in the adjacent jet facility. New proprietary concepts in transmitting color separation, light collection, and novel prism separation of the scattered light have also been designed and built into the system. Off-axis beam traverse and alignment complexity led to the requirement for a specialized, programmable traverse controller and the inclusion of an additional traverse for the off-axis arm. To meet this challenge, an "in-house" prototype unit was designed and built and traverse control software developed specifically for the water tunnel traverse applications. A specialized data acquisiton interface was also required. This was designed and built for the Laser Fluorescence Anemometer system.

14. SUBJECT TERMS
Laser Velocimetry
Water Tunnel Measurements

15. NUMBER OF PAGES
91

16. PRICE CODE
A05

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102